AN END-TO-END FREEZE TCP FOR AD-HOC NETWORKS

BY
SUNG RAE CHO

Supervisor
Associated Professor H.R. Sirisena

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering
In Electrical and Computer Engineering
At University of Canterbury
Christchurch New Zealand
March 2004

ABSTRACT

SACK TCP, as the baseline, performs promisingly in the wireline network where major control point is of the network buffer utilization. Basically, the reactive based congestion window progression causes high network buffer utilization due to the likelihood of large bursts of data, and it deteriorates the network bandwidth utilization: the resultant end-to-end RTT inflation results in large retransmission timouts (RTOs), thus long timeouts, and reduced TCP throughput.

In this thesis, we have evaluated that through a series of simulations performed in ns-2, the baseline, of such large bursts of data transfer and the resultant large RTOs, is not suited for ad hoc networks in terms of power and badwidth limitations. We investigated a receiver-oriented rate controller (*rater*) dominated by the resource-constraint ad hoc links, where the link availability changes not gracefully but transiently.

Assuming that end clocks are both synchronized in a passive or active way and TCP takes into account the TTL field of the IP header each time when data packet is received, the forward link delay (FLD) and the current hop length give the instantaneous throughput available. Then, in order to throttle the sender' congestion window by using the advertised window field, the receiver employs the *congestion window delimiter* that is characterized by the 802.11 MAC protocol.

In addition, the transient nature of the medium availability due to medium contention proposes the freezing timer (*freezer*) to be equipped at the receiver-end that periodically freezes the sender in cases of heavy contention present. In this sense, two other metrics, i.e., *buffer occupancy* and *contention factor* have been introduced to perceive the degree of the buffer utilization and the medium contention, respectively, for supporting the *delimiter* and the *freezer*.

Finally, an elaborate sender-end, namely *ad hoc sender enhancements*, was proposed for achieving the optimized behaviors of the receiver-end enhancements, as an optional deployment. It implemented an add-on probing mechanim to deal with the route disconnection problem by means that the probing backoff supersedes the RTO backoff. The combination of the schemes, i.e., *delimiter + freezer + ad hoc sender enhancements*, is called the *ad hoc TCP*. It outperformed the baseline in perspective of both throughput and, especially, *goodput.* The primary merit of our *ad hoc TCP* is that such propositions are based on solely end-to-end, so do not require the network-originated feedback.

Acknowledgements

I would like to acknowledge precious people who helped me accomplished the thesis work with expected input. I thank my supervisor, Associate Professor Harsha Sirisena, for accepting me into the master's programme and, during the research period, giving me enough time for my individual research. Especially, I thank my co-superviosr, Associate Professor Krzysztof Pawlikowski for approving, and proof-reading, my thesis. I would also like to thank, Gustavo Da Costa who provided useful simulation scripts and codes during my early brainstorming period to find a feasible solution, and Hyun Park who presented useful comments and information whenever I encountered unforeseen problems. I also express my gratitude to Insung Hwang who steadily encouraged me to keep up working and told about the importance of the attainment of M.E degree and in addition, to all the other fellows in the Network Research Group. Finally, I sincerely express my appreciation to my beloved parents, brother, and specially my aunt who have all constantly inspired and assisted me.

List of Contents

List of Figures

List of Tables

List of Equations

Chapter 1
INTRODUCTION

1.1 Intoduction to Thesis topic

The services offered by 3G wireless networks to mobile users have created a huge volume of network traffic. This trend of increase in network traffic will continue in future due to an ever increasing number of mobile users. Also recent developments in the mobile technology have spurred a great enthusiasm among its users for a rapid adoption of various new functionalities, such as full multimedia services requiring high bandwidth and high data transmission rates. Furthermore there is a rapid progression towards an all IP based 4G network systems which can offer a global connectivity between end users.

Presently mobile devices in use include cellular phones, PDAs and laptops equipped with radio transceivers. All of them allow mobile users to communicate with each other in portable environments of heterogeneous networks. However, such mobile and portable devices rely on existing network infrastructures, such as routers, base stations and access gateways, for seamlessly connecting with each other.

The recently growing trends in demand of mobile computing devices promote ubiquitous networking that does not require the use of pre-existing infrastructure. In rescue situations after such natural disasters as earthquakes, or in specific congregation situations like conferences and concerts, well-equipped network infrastructures may be not available. In such circumstances a temporary network, which is capable of meeting the basic communication requirements, will be needed.

Such a standalone network would be implemented as mobile wireless ad hoc network, MANET (Mobile Ad-hoc NETwork) [45, 98]. It forms a challenging research area as can be witnessed in recent networking literature and conferences.

1.1.1 Ad hoc TCP in use

Ongoing research in the area of ad hoc networks has focused on various aspects of routing protocols, due to dynamic nature of the topology. A related research topic is on the Medium Access Control (MAC), to fairly share the single common wireless channel.

After incorporation of MAC and routing protocol suite, employed to offer bandwidth to the transport layer, researchers have investigated a transmission control protocol which can fully utilize the available network resources, shared with fairness among its users.

Many schemes recently proposed for ad hoc networks and based on Transmission Control Protocol (TCP) deal with the route disconnection problem (each time when a hop link is broken or about to be broken) by placing the underlying TCP mechanism into the persist state (*frozen state*) until a new link is restored. If no such freezing (*snooze*) state exists then TCP sender will suffer from a number of unnecessary slow start phases which will waste the network resources. It will also suffer from a spuriously diminished slow start threshold (*ssthresh*) because after a new route is restored it would not be retrieved plausibly. As a solution to the route disconnection problem, Explicit Link Failure Notification (ELFN) approach, to the support of inter-layer information delivery from MAC protocol has been proposed in [74]. It is a promisingly flexible and sustainable remedy for such problems.

However, few TCP schemes attempt to optimize the transmission rate according to the bandwidth available under a random access MAC protocol, given that routing protocol maintains the path link connectivity. By definition, mobile nodes compete for a single channel available; therefore, bandwidth available per node is highly fluctuating during communication. In the case of intensive medium contention occurring in the middle of the path link, the network-supported scheme can be one of the solutions to mitigate the contention degree. It will freeze the sender instantly or reduce further transmissions. Yet, no complete end-to-end approach has been proposed. As a consequence, we investigate an end-to-end TCP scheme to control the bandwidth-constraint ad hoc links without the explicit signaling from the network.

1.1.2 Current TCP sub-optimality

Once a TCP connection is established in an ad hoc network, participating intermediate nodes (on the path link between end nodes) are all freely moving, so forming a dynamically changing topology during the whole connection time. A routing protocol is thus employed and plays a role in the route maintenance as well as its discovery. Meanwhile, the common medium should be shared fairly, according to a MAC protocol.

In an ad hoc network, mobile nodes that employ a particular pair of the lower layer protocols will require a transport protocol. However, the current TCPs for either wireline or wireless networks cannot be directly employed for ad hoc networks as they are not suitable for such networks due to the following major reasons.

(1) The wireline TCP is optimized for buffer related packet loss for use over fixed wireline networks, as TCP sender quenches its transmission window drastically each time a packet loss occurs. On the other hand, the ad hoc networks are not likely to suffer from the buffer overflow but rather from the link-related errors: such as collisions in MAC layer (namely, MAC collisions), wireless link corruptions[1], routing failures, and other reasons. Therefore single wireless link related errors need not shrink the transmission rate drastically unless burst wireless errors exist. (2) The wireless TCP uses a centralized base station which supervises connections made between end nodes lying in two different networks (as the *split* TCP connection[2] does). Therefore, both types of TCP cannot be used without a proper modification.

As a proposed modification, TCP end node should take into account some useful information which can be obtained from the lower layer protocol specifically deployed. We will assume that TCP is aware of link conditions (i.e., medium contention and link breakage) to the extent the lower layers can determine by themselves.

---

[1] Bit error rates in wireless communication channels are in the order of 10^{-6} or even worse if a link layer protocol is not used as opposed to 10^{-12} in wired. Thus, the link layer acknowledgements are necessary for better reliability, to mitigate wireless link related packet losses.

[2] The split connection (e.g., I-TCP [14], M-TCP [38]) or proxy (Snoop [19], WTCP [126] as a TCP aware link layer scheme) approaches require an intelligent base station between end hosts and so are not suitable to provide a proposal for ad-hoc networks because ad-hoc networks are without fixed base stations between hosts.

Our proposal addresses the following three issues:

- In case of packet delay induced by heavy medium contention, we propose a technique for mitigating the degree of the contention by reducing transmission rate. It should reduce the impact of packet delays because extremely inflated degree of the contention can cause premature retransmission timeouts (RTOs).
- In case of wireless link errors, the sender should not shrink its window size to a small value if it was a transient wireless error [1], and
- Finally in case of sudden link breakages, the sender should not suffer from the TCP exponential backoffs[2] [5, 109, 138].

1.1.3 Desirable TCP enhancements

In today's Internet, there are many TCP variants working for end-host flow control. Each TCP has its specific congestion control and congestion avoidance algorithms. These algorithms are designed to react in the case of network congestion between end hosts. Whereas the routers have a particular queue management algorithm that characterizes packet dropping probability at times of impending congestive stage or buffer overflow. They usually play important role in indication of congestion by dropping or marking the packets (as congestion approaches), thus providing a necessary feedback to TCP senders for reducing their sending rates.

For marking the packets the routers usually employ Explicit Congestion Notification (ECN) for end-to-end congestion control in present wireline TCP networks. They send congestion information to end hosts about incipient congestion by marking bits in received packets. This congestion information is ultimately received by end host through the ACKs.

However, we are not expecting such a dedicated network support in mobile networks, because of scarcity of network resources such as bandwidth and power, and rely on end-to-end enhancements which complement the end-to-end performance of TCP. The receiver-dedicated modifications must

[1] Coupled with a link layer scheme to improve the hop-by-hop reliability, wireless link errors are recovered, inducing considerable delays.

[2] Dyer et al. [49] simply propose the use of fixed RTO: the loss of packets is caused by frequent route disconnections, due to high node mobility as well as network congestion. Thus, for the high node mobility environments, the RTO remains fixed until the route is restored and the retransmitted packet is acknowledged. However, a fixed RTO may not be adequate for an extremely long route between end hosts. The choice of a fixed RTO should be flexible according to the hop length of the path link.

be favored because a nomadic node of an ad hoc network can be more likely to behave as a receiver and can communicate with a wireline IP network which is difficult to modify.

1.1.3.1 Interoperability with existing networks

Due to growing demand for accessing to the Internet, ad hoc mobile users should be able to be linked with wireline IP networks. Mobile IP technology presently allows the mobile nodes to communicate with wireline IP network or the Internet (it requires appropriate address management and protocol interoperability particularly due to heterogeneity of routing functionalities [72, 116, 127]). However, there exists a variety of end hosts employing different, incompatible transport layer protocols. Thus, the interoperability issue against existing networks needs to be solved. The aspect of the compatibility among the wireline and ad hoc networks has been suggested as a future direction of work.

1.2 Research Objective

The objectives of the thesis are:

- To survey the basic functionalities and potential shortcomings of the MAC protocols used for ad hoc networks.
- To survey the existing TCP schemes for an ad hoc networking environment.
- To investigate key requirements for efficient TCP modifications for ad hoc networks. In particular:
  - High Packet delivery ratio, for achieving power and bandwidth effectiveness.
  - No exponential RTO backoffs, by means of the sender's probing capability. The probing backoff superseding the current RTO backoff is considered.
  - Significant reduction of the number of premature RTO expirations.
  - Fast adaptability to hop length changes—each time hop length changes, the receiver should renew the contingent measurements.
  - Expected global fairness, achieved by identical *raters* deployed (i.e., the sender will be generally *receive window-limited*).
  - Develop a scheme that does not require changes to the fixed end-host when connected to a wireline network.
- To verify the performance achieved by the proposed scheme and compare it with the baseline TCP (i.e., TCP SACK).

- To discuss the deployability of the proposed scheme in the wireline network and the ad hoc-cum-wireline network.

1.3 The layout of the thesis

This section provides a brief overview of the layout of the thesis.

Chapter 2 explains the structure of the protocol stack for the ad hoc mobile node and the basic characteristics of an ad hoc network. It then provides a detailed explanation of MAC protocol (Chapter 2.3) as well as transmission protocol (Chapter 2.4). In addition the current state-of-art TCP schemes (Chapter 2.5) for ad hoc networks and other possible end-to-end schemes (Chapter 2.6) are discussed.

Chapter 3 then details the rationale of our newly proposed TCP scheme and elaborate about receiver (Chapter 3.3) and sender (Chapter 3.5), providing some prerequisites in Chapter 3.2.

Chapter 4 presents the experimental methodology used by the network simulator to perform the simulations. We also discuss how the ad hoc sender and the receiver were implemented.

Chapter 5 assesses our propositions in several different paradigms, such as stationary and dynamic moving topologies, in variation of traffic load and mobility.

Chapter 6 overviews results, draws conclusions, and provides a discussion of future work by addressing potential issues envisioned for ad hoc networks.

Chapter 2
Mobile Ad hoc networks

This Chapter will present general ad hoc characteristics and enumerate some useful imformation related to protocol stacks. Then, general issues in each of primary, resource-dominiating, *ad hoc tiers* (i.e., MAC, Routing, and Transport protocols) are followed up.

2.1 The MANET Protocol Stack

In networking literature the model derived from combining different layers is referred to as a *protocol stack* (e.g., ISO-OSI, TCP/IP, and customized user model). The widely deployed TCP/IP protocol stack, at most of end nodes, has been defined in layered manner and well documented in RFCs[1]. Each layer has been carefully defined and developed over many years to provide a stable and robust platform for implementation and for further development.

As shown in Figure 2-1, the MANET protocol stack is similar to the TCP/IP suite, but also has held its own characteristic protocol suite, such as MAC, routing, and transport protocols. It shows a support of a mobile node, provided by the CMU monarch's wireless extensions, for accurately simulating multihop wireless ad hoc networks. It built a particular network stack of the mobile node. The network stack consists of a link layer (LL), an ARP module connected to LL, and interface priority queue (IFq), a MAC layer (MAC), and a network interface (netIF). The network interface plays a role in serving as a hardware interface to access the channel, and the model approximates the Lucent WaveLan DSSS (Direct-Sequence Spread Spectrum) radio interface, which is a commercial radio interface card. A more detailed description related to simulation aspects can be found in [37].

[1] Request for comments—of all standards related to TCP/IP, which are done by the IETF. RFCs are the most easily obtained and widely available in the public domain http://www.rfc-editor.org/rfc.html [Accessed from 5 Jan 2004].

The well-known TCP/IP protocol stack is a five-layered model. At layer 1 the unit of information is called a *bit*, at layer 2 it is a *frame*, at layer 3 it is a *datagram*, at layer 4, it can be either a *segment* or a datagram, and at layer 5 a *message*. In general, IP datagram traversing through networks is called a *packet,* RFC 1122 [35].

The layers 1 and 2 of the TCP/IP model are not actually defined by the RFCs, as the TCP/IP was designed to be independent of physical media. In other words, it can operate over any network that can move bits from one place to another with a reasonably low bit error rate.

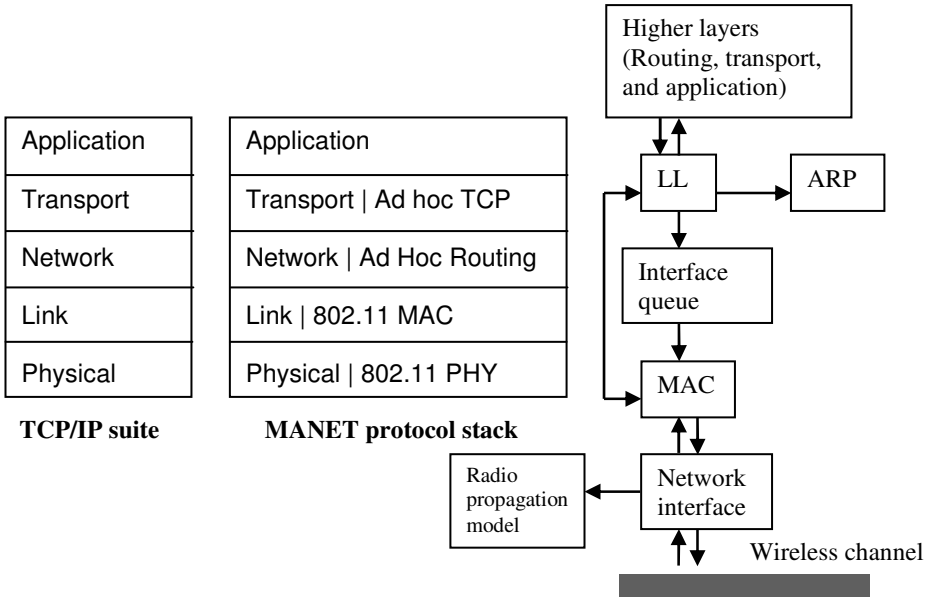


Figure 2-1. Protocol stack models and schematic view of a mobile node

Depending on the type of network, different physical equipments or protocols might be used. For instance, in the physical layer particular equipment may be used: such as copper wire, optical fibre, and wireless transceiver. In the MAC layer MAC protocol can be effectively configured to share the common medium (e.g., Ethernet, Token Ring, CSMA/CA and Frame Relay etc). Also the link layer protocol provides the hop link reliability in support of ARQ, FEC [40], or other link level protocols (e.g., [18, 113]). In addition, link layer protocols implement an ARP (Address Resolution Protocol) module, which exchanges ARP packets providing mapping between IP addresses and MAC layer addresses. It also deals with fragmentation and reassembly of packets.

For an ad hoc network, the mobile nodes are equipped with a specific standard for wireless channels. Examples of such standards are IEEE standard for wireless LANs, IEEE 802.11, European standard for high speed wireless LAN, HIPERLAN 2 and an industrial approach, such as Bluetooth.
One of the most prominent traits of ad hoc networks is that nodes can move around freely, thus they can move out of transmission range of each other. A routing protocol is employed at the layer 3 *ad hoc routing* and should play a vital role in maintaining the link connectivity of each other as well as discovering a route, whenever necessary.

A number of routing protocols have been designed, each with their own merits, such as general shortest path routing, longer but stable routing [51], and MAC aware delay-oriented shortest path routing (DOSPR) [146]. In particular, DOSPR argues the fact that MAC availability dominates routing functionalities because the routing procedure needs to flood control packets and because it can be possible only when the medium is available even if routing packet is of high priority. Several papers, e.g. [54, 128], overview the state-of-the art routing protocols and carefully enumerate them.

DSR [80] and AODV [52, 117] are two popular routing protocols which are being used for studying most of ad hoc networks. Both are reactive protocols and require a node to initiate a route discovery procedure to find a route to its destination. DSR finds the complete path to the destination, whose overhead is included in every data packet, while in AODV the nodes along the path only know about the next neighbor for the packets to be forwarded. By relaying packets neighbor-by-neighbor they will reach the destination node. Thus, DSR needs more overhead while sending data and AODV needs more overhead while performing route discovery.

The amount of routing overhead is differentiated by the routing protocol employed, such as source initiated on-demand or table-driven. Observations made in [125], which evaluated the performance of various TCPs in particular over static ad hoc cases, imply that the required routing overheads differ according to the kind. So, the preferred routing protocol to select is subject to diverse situations, such as mobility and traffic load, and takes an effect on TCP performance differently. In case of high mobility or frequent route breakages, the amount of local retransmission attempts and routing control packets are likely to increase resulting in a longer end-to-end delay in path link (links between source and destination). In the meantime, the TCP retransmit timer may expire and so invoke unnecessary congestion control algorithm; moreover, successive timeouts unreasonably reduce the slow start threshold (*ssthresh*) inadequate for the new reconnected route, and so underutilize network resources in a non-trivial extent. Hence, it turns out that, an employed routing

protocol influences to the TCP performance considerably in terms of the expense of the functionalities performed [49, 125]—there is no doubt that TCP-end should throttle transmission rate by itself each time a considerable delay is perceived at either TCP-end.

At layer 4 there are two options as the most common transport protocols, the User Datagram Protocol (UDP) [1] and the transmission Control Protocol (TCP), but other protocols are also used. UDP is simple enough that there is only one version in use but TCP has continually been modified and there exists many different versions of it.

Finally, layer 5 is the application layer. This layer provides services suitable for the different types of application that might use the network. For example, there are terminal connections (Telnet), electronic mail, the Simple Mail Transfer Protocol (SMTP), the File Transfer Protocol (FTP), DNS, NTP (Network Time Protocol), and NFS.

2.2 General Ad Hoc peculiarity

This chapter introduces the general peculiarities of ad hoc networks. Each contemplates key requirement for the proposed TCP scheme.

2.2.1 Bandwidth-limited shared medium

Mobile nodes share a common medium to communicate with each other in a way of random access basis. Thereby, a number of MAC collisions might occur in the case of highly competing environments.

In addition, many other factors (such as, physical obstacles, noise and interference by others and vulnerable link sustainability) may apply to drastically restrict the available raw bandwidth, compared to wireline links; therefore, the available bandwidth is highly oscillated and is usually never fully utilized the *raw* bandwidth[2]. In this sense, in the case of communication with a wireline

[1] The Use Datagram Protocol [121] is a very simple transport protocol. There is no guarantee that the data is ever delivered to the destination and the date packets can arrive in any order. 802.11 RTS/CTS exchange and link level ACK protects UDP packets from collision loss of the hidden terminal problem and wireless link corruption, respectively. Since there is very little overhead to each packet, the simplicity makes it faster than more complicated protocols like TCP.

[2] Each radio interface is identically equipped for use and bounded to the same maximum raw line rate that gives a range of allowable changes in capacity of a single node (Chapter 3.4.6.1).

end host, resource-dominating ad hoc links are of primary concern to determine an allowable transmission rate. A more detailed explanation of MAC related concerns is found in Chapter 2.3.

2.2.2 Power-limited mobile nodes

The processing capability of mobile nodes is restricted because of a limited battery power. This constraint requires low processing overheads of nodes. Inter-router exchange and network based notification of control packet is thus highly prohibited. In particular, if dedicated routers functioning flooding control algorithm or QoS mechanism [1] are not available, intermediate routers might not want to consume considerable power for serving others. In this sense, network-originated control beacons and session-related states maintained in the network might not be desirable, which thus encourages a solely end-to-end based TCP scheme.

2.2.3 Network partitions

Since there is no central base-station in the ad-hoc network structure, it is likely that one network becomes entirely partitioned from the other part of the network. If the sender and the receiver of a TCP connection lie in different partitions, transmitting packets get dropped by the network, resulting in the invocation of the congestion control algorithm. Then successive failures (timeouts) in the slow start phase will shrink its slow start threshold (*ssthresh*) unreasonably and in turn underutilize available network resource, where no explicit end-to-end scheme is available for restoring the unnecessarily diminished *ssthresh*—For instance, ATP [139] requires network-originated feedbacks to fast restore, and TCP-westwood 99] mitigates the impact of spuriously reduced *ssthresh* by means of end-to-end bandwidth estimation but cannot be directly employed to the ad hoc network.

2.2.4 Multipath routing and Frequent route changes

Multiple routing is commonly utilized in ad hoc networks, thus sometimes degrades TCP performance because TCP end considers the sequential order of packets for sizing the congestion window. For instance, a routing protocol, such as TORA (temporally-ordered routing algorithm),

[1] In general, the IntServ approach is undesirable for a power-constrained ad hoc network because it requires power consuming, high processing overheads in order to function its basic components such as RSVP, admission control, classifier, and packet scheduler. Intuitively, the light weight DiffServ approach is preferred in the ad hoc network.

maintains multiple routes between source destination pairs, in order to minimize frequent route recomputation. It might iteslf result in a large number of out-of-sequence packets arriving at the receiver. In response, the receiver generates duplicate ACKs for every out-of-sequence packet. This causes the TCP sender to invoke the congestion control algorithm unnecessarily. Spurious reductions made in the congestion window and *ssthresh* should be avoided.

As a matter of fact, the sender might not completely prevent spurious fast retransmits due to inevitable high out-of-order delivery rate in terms of the dynamic ad hoc routing. To a certain extent, Eifel algorithm, TULIP, or D-SACK (detailed in Chapters 2.6.3, 2.6.4, and 2.6.9, respectively) endeavors either to alleviate or avoid the impact of spurious fast retransmit, but considerable expense must be necessary.

As in our proposal, TCP ends may be able to be robust against the spurious fast retransmit problem. To achieve this, the receiver informs the sender of an estimated slow start threshold periodically, and in turn the sender will not shrink the congestion window drastically but to the estimated threshold; in addition, this can be useful in case of timeouts, because the sender may also spuriously quench *ssthresh*.

In terms of route change due to node mobility, this changes very often during the lifetime of the connection. It is noticed that when node mobility increases, route breakage and formation become more frequent. So, it increases local retransmission rate, and more overheads are required for flooding control packets. More importantly, the congestion window computed for one route may be too large for a newer route and vice-versa. Hence, the congestion window should be recomputed or properly set with an appropriate *ssthresh* after reconnection (e.g., TCP westwood [99], ATP [139]); otherwise, it is likely to overwhelm or underutilize the network capacity—for instance, resulting from a high set of *ssthresh*, the fast increase of the congestion window may be harmful in that it might hamper the transmissions of adjacent nodes, and vice versa.

In brief, TCP needs to be aware of path rerouting in order that the congestion window is set appropriately according to the rerouted path condition. The proposed TCP scheme will modify the receiver to be able to estimate *ssthresh* representing the current medium contention degree of path link. This is useful for eliminating the impact of spurious fast retransmits and timeouts.

2.2.5 High bit error rate

Higher bit error rate of error-prone wireless link is commonly in the order of 10^{-6} or even worse. Each node should thus definitely require reliable link layer protocol(s) to assure the hop link delivery, which then improve the end-to-end throughput. However, the use of a link layer scheme can cause inflation of round trip time and then lead to spurious timeouts. S. Floyd [62] and R. Ludwig [94] introduced DSACK and Eifel algorithm, respectively, to address this problem (likewise as with the reordering or rerouting problem as mentioned in the previous section). The DSACK and Eifel algorithm can, in view of transport layer, mitigate the impact of spurious timeouts by undoing the reduction made. The detailed descriptions of the DSACK and the Eifel algorithm are found in Chapters 2.6.9 and 2.6.3, respectively. On the other hand, R. Ludwig [92, 93] proposed effective retransmission timers (related to Chapter 2.6.8), which are robust to spurious inflation of the round trip times of packets. This could alleviate the spurious RTOs to a certain degree. The support of the complementary algorithms can further encourage the deployment of more reliable link layer protocol(s) responsible for avoiding per-hop link errors.

2.2.6 Requirements for Ad Hoc TCP

As a result, specifically, TCP modifications can be confined at receiver side but extended to the sender side as well.

- The receiver feedbacks *ssthresh*, which determines the network path link capacity to an extent.
- The sender in response should be modified to fully utilize the receiver's feedback information so that it can perform promisingly in case of the following events, such as:
    - Medium contention-induced blocking and sudden route disconnections causing spurious timeouts, and
    - Instant route switches and wireless link corruptions causing fast retransmits (a non-sensible reduction of *ssthresh*).
- If the sender or the receiver in ad hoc networks communicates with an unmodified wireline opponent, it should behave differently (Interoperability aspect).

2.3 Ad hoc MAC protocol

In multi-hop ad hoc networks, nodes communicate with each other, sharing a common wireless channel, and thus require a fair, or an effective, MAC scheme to occupy this shared medium. This Chapter introduces several random access based MAC protocols, then explains how they work and what deficiencies each of them has, and finally provides the awareness of how TCP performance is restricted by the MAC efficiency.

Numerous MAC layer protocols have been proposed aiming at different aspects of achievement and different types of network for deployment, yet, a few of them have been designed for a multi-hop wireless link, and a few of them have been evaluated for use in the multi-hop ad hoc network.

Several popular random access wireless MAC protocols, which were firstly developed for wireless LANs, have been addressed in [142] and provided benchmarks of design choices of the protocols when used in ad hoc networks, e.g., Carrier Sense Multiple Access [1](CSMA), Multiple Access with collision Avoidance (MACA) [82], MACAW [27] as an enhancement of MACA, Floor Acquisition Multiple Access (FAMA) [59] and IEEE 802.11 [75]. M. Gerla et. al. [65] evaluated the performance of the MAC protocols in terms of TCP throughput. In particular, due to the fundamental competing situation of the common path between TCP data and ACK flows (by the preference of most routing protocols) when having a window greater than one packet, MACAW outperforms the others owing to the benefit of the additional control frames (DS, ACK, RRTS) [2] as well as MILD (Multiplicative Increase Linear Decrease) backoff policy that achieves better fairness compared to the aggressive random binary backoff. Particularly the addition of link level ACK in MACAW precludes wireless hop link error.

At present, as a standard gradually evolved and currently adopted for wireless ad hoc networks, IEEE 802.11 (802.11 for short), equivalent to MACAW, has been studied, addressed and developed in many research papers. K. Tang and M. Gerla [143] discussed CSMA, FAMA, and 802.11, and evaluated under a string, ring, and grid topology in order to verify their prominent features. CSMA

[1] CSMA undertakes carrier sensing before transmission in order to occupy the medium. When the channel is free, a packet can be transmitted, otherwise it is rescheduled after a random timeout. As the major limitation, CSMA suffers from the hidden and exposed node problem.

[2] Complementing the RTS-CTS control frame exchange, DS frame is in turn sent immediately followed by data frame. After that acknowledged, RRTS, in response to another RTS, will be sent back to inform the time when the link becomes idle. Afterwards, the regular frame exchange process begins immediately.

performed best only when there were no competing TCP sessions while FAMA worked well except when mobility was applied. They concluded that 802.11 would provide a superior combination of fairness and aggregate network throughput under topologies being studied, while CSMA and FAMA specifically suffered, as expected, from the hidden terminal experiment and the situation where mobility was applied, respectively. Besides, 802.11 mitigates the unfairness problem to a considerable extent in a contending environment.

2.3.1 Negative aspect

Although the 802.11 MAC scheme as a proposed standard has been adopted for most ad hoc studies and evaluated to give the best promise in most ad hoc situations, interestingly D. Dmitri et. al. argued in [118] that, 802.11 suffered from a high number of CTS (Clear-to-send) timeouts in cases of frequent topological changes (thereby reducing the overall throughput) and occasionally CSMA consistently outperformed 802.11 when nodes were mobile. The reasoning behind this is that in highly contending moving nodes being present, RTS-CTS handshakes failed many times and wasted the substantial amount of scarce bandwidth. Thus traditional intrinsic carrier sensing access will be more beneficial to attain relatively high TCP throughput.

Moreover, unfairness issues related to the capture effect and the well-known terminal problems among a number of competing nodes were addressed in [65, 142, 143], where authors proposed that they could be mitigated to a certain degree by means of properly increasing the Inter Frame Spaces[1] (IFSs) intervals or more intelligent back off schemes enlarging the medium idle period between frames. However, S. Xu and T. Saadawi [153] addressed the limitation of the variation scheme of the IFSs [142, 143] by a shift to improve the unfairness because the IFS variations substantially causes the aggregated throughput to degrade so badly.

2.3.2 Need of QoS MAC

As potential, unsolvable problems in such a random accessing MAC scheme, over many competing nodes, a substantial number of collisions due to the well-known MAC problems (802.11 never eliminates those problems because of the imperfection of the random accessing scheme) suffer from plenty of back offs. It seems unacceptable in critical environments where multiple TCP flows on a

[1] Inter Frame Space (sometimes called yield time) defines the amount of time the sender backs off before sending another frame after transmitting a frame.

single node are common, and the other nodes being blocked might result in serious problems in search, rescue or military purposes. Therefore, more research is anticipated to develop a new MAC scheme, so that a transmitting node will not be interrupted by the transmission of another, and in turn to provide a guaranteed bandwidth.

Numerous QoS MAC protocols have been developed to support the guaranteed bandwidth for realtime services, eliminating the well-known MAC problems.

FPRP (Five Phase Reservation Protocol) [159], HRMA (Hop Reservation Multiple Access) [140] and CATA (Collision Avoidance Time Allocation) [141] have been developed to provide time bounded services as channel occupation, but potentially there exist considerable limitations due to the lack of the centralized administration that assigns dynamic slot reservation on demand when deployed in an ad hoc situation.

SRMA/PA (Soft Reservation Multiple Access with Priority Assignment) is cited in [1] that allows nodes to carry two types of traffic differently according to priority level, such as urgent realtime data (to take the medium from other nodes), and no-realtime data (on a demand basis).

CTDMA (Cluster Time Division Multiple Access) as in [145] was designed as a spatial channel reuse strategy to dynamically partition the network into clusters where each cluster uses a different DSSS (Direct Sequence Spread Spectrum) spreading code for identifying channels (CDMA facilitates various connections with different codes in the same time slot.) and additionally uses a unique, globally synchronous slotted TDM frame, and leftover free time slots are occupied in a random access basis.

MACA/PR (Multiple Access Collision Avoidance with Piggyback Reservation) [90] is an extension of 802.11 and FAMA and is able to guarantee reserved bandwidth for real-time traffic by means of the TDM based reservation carried by real-time scheduling information at the headers of data and ACK frames.

Such network-wide timing synchronization and code separation supporting the realtime traffic can definitely improve throughput significantly owing to spatial channel reuse. However, it seems to be impractical to implement multiple receivers that can receive multiple codes simultaneously and

infeasible to require a substantial effort in the global synchronization at critical situations because of imposing complexity and huge expense.

The following subchapter introduces more aspects about the 802.11 medium access way and presents its potential problems and strategies to mitigate those problems in more detail.

2.3.3 IEEE 802.11 framework

As the standard MAC layer protocol adopted and widely used in testbeds and simulations for wireless mobile ad hoc networks because of its simplicity of implementation, cost effectiveness and prevalent availability, 802.11 [46, 75] MAC protocol provides an efficiently shared, contention-based broadcast channel.

As the evolution of random access MAC schemes, whose fundamental medium access mechanism is called distributed coordination function (DCF), 802.11 incorporates CSMA with Collision Avoidance (CA), link-level ACKs, and the virtual carrier sense mechanism, which employs the RTS/CTS control frame exchange for channel reservation.

In fact, this protocol was not designed for the direct use in the multi-hop based ad hoc networks. Several research papers [25, 128, 142, 143, 153, 157] addressed potential medium contention problems, resulting from the well-known difficulties as briefly mentioned before, proposing not perfect but reasonable solutions.

2.3.3.1 Medium access

In 802.11, the DCF needs two basic Inter frame Spaces (DCF Inter Frame Space (DIFS) and Short Inter Frame Space (SIFS)) that support asynchronous data transmission with the provision of different priorities in the random accessing manner. Every link data frame to be sent will include a sequence of RTS-CTS-DATA-ACK frames exchanged between one node and another, whose time period means the completion of one data transmission. For transmitting the full sequence between the nodes, the other nodes that can hear it by means of a virtual carrier sense mechanism detecting when the medium is busy, should postpone their transmission accordingly by means of updating a Network allocation Vector (NAV), which contains the information of the period of time for which the channel will remain busy.

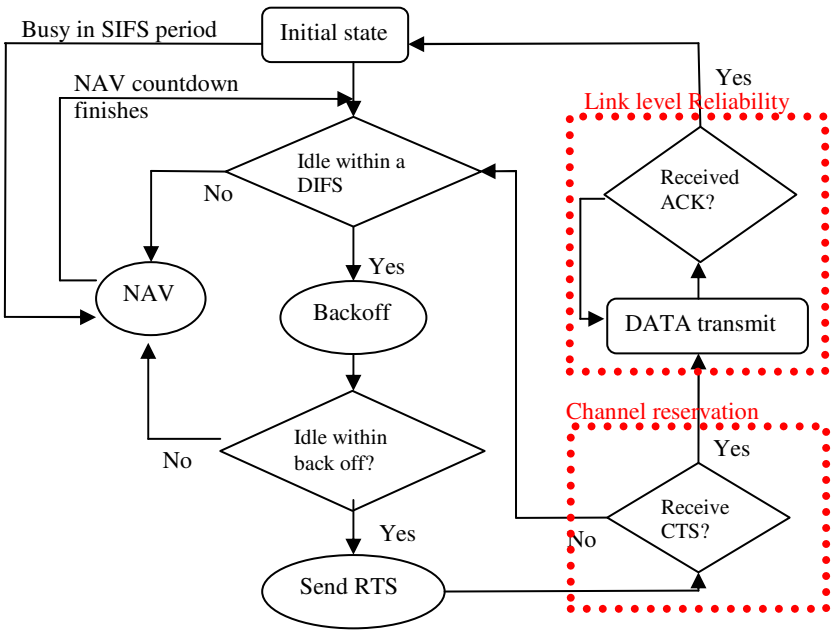


Figure 2-2. The IEEE 802.11 framework to access channel in a random basis.

In other words, in the case of one node transmitting data to one another, the sender eavesdrops to check whether or not the medium is idle. Afterwards, it tries to exchange RTS-CTS control frames in order to avoid the well-known hidden terminal problem.

The successful RTS-CTS exchange must assure the non-existence of any other nodes which are out of carrier sensing range but want to transmit at the same time. When many competing nodes are present to occupy the medium, each node should await an additional backoff period to sustain transmitting even though the medium is idle in the DIFS period because the transmitting node wants not to fail its control packet exchange, also not to hamper other communications of adjacent neighbors. Specifically, the back off period, such as the random binary back off, or the MILD back off scheme adopted in MACAW, will mitigate the likelihood of undesirable capture effect[1] and decrease collisions owing to different remaining back off periods as in Figure 2-3. After the completion of the RTS-CTS exchange, the node is allowed to transmit data. An intermittent time period, the SIFS (Short Inter Frame Space) period, immediately before each of CTS, DATA and ACK frame, is shorter than the DIFS and so assures of no other nodes attempting to transmit within

[1] In the 802.11, as addressed in [142, 143], fairness improvement against the capture effects among simultaneous TCP connections can be made by simply adjusting the back off schemes and IFSs parameters at the expense of reduction in aggregate throughput. Further research [109, 153] addressed serious problems encountered in the 802.11 based ad hoc network and concluded that the current version of 802.11 does not function well in the ad hoc network in certain cases.

this period—SIFS implies that CTS, DATA and ACK frames have a higher priority than RTS frame. Figure 2-2 illustrates the 802.11 channel reservation mechanism.

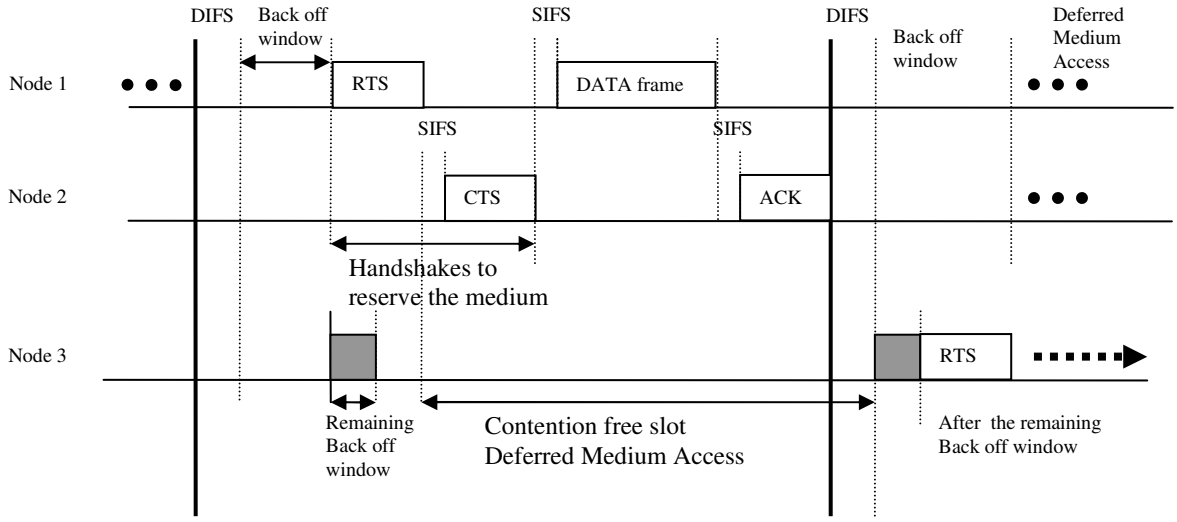


Figure 2-3. Random medium access with differentiated back off periods (If unsuccessful attempts, the back off window will be enlarged to reduce the likelihood of collisions.)

2.3.3.2 Drawbacks

In the context of the random access based 802.11 MAC protocol, the more wireless nodes that exist in the interference range, the more contention takes place. The study in [129] presented that the current implementation of the RTS/CTS mechanism may cause RTS/CTS-induced congestion among competing nodes to an extent that any node becomes unable to transmit any packet during long periods of time, which manifests itself in the form of congestion. Such spurious RTS/CTS mechanism, undoubtedly, degrades TCP throughput. Thus, in terms of the throughput and the power effectiveness, the RTS/CTS mechanism has to be tuned more intelligently (e.g., by the awareness of (sporadic) transmitting nodes being out of carrier sensing range, but capable of potentially interfering transmissions). In addition, Jinyang Li et. al. [88] detailed that the backoff mechanism of 802.11 unnecessarily wastes considerable time when forwarding packets in a contending environment, and it is problematic for it to achieve the ideal throughput.

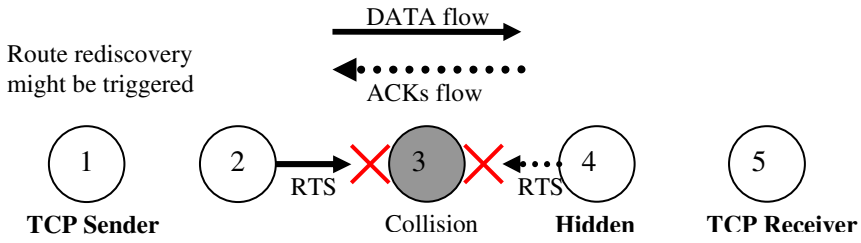
2.3.3.3 TCP interaction with MAC protocol

As illustrated in Figure 2-4, in view of TCP flow where, due to the preference of most routing protocols, the forward data flow is likely to be equivalent to the reverse link of ACK flow, the hidden terminal situations will derive a number of unsuccessful attempts of RTS-CTS handshakes

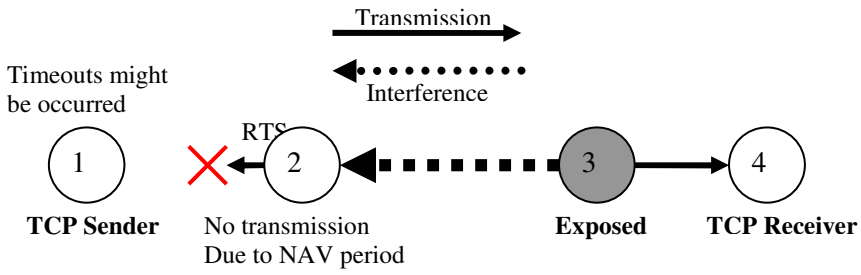
and cause prolonged contention-related delay (i.e., substantial back offs), implying a number of consecutive NAVs (i.e., long NAVs), which result in undesirable performance degradation.

When a hidden node condition as in Figure 2-4 (a) takes place, due to extremely high traffic from node 4 to node 3 of counter-flows of ACKs[1] in this case, node 2 may be likely not to succeed and probably will drop the packet after a maximum MAC retry attempts. It then tries locally to restore the route or sends back a route failure message to node 1, so as to invoke the route discovery procedure. In the meantime, occasionally, node 2 drops all packets queued (if it does not keep them for a certain time) because of a currently broken link.

Likewise, in the case of the exposed terminal problem in Figure 2-4 (b), the interference of node 3 will disable node 2 to even trigger RTS-CTS handshakes to occupy the medium, placing to long NAVs. It results in pending TCP ACK flow and, in turn, TCP timeouts at the TCP sender. Unfortunately, the TCP sender will then back-off its RTO, and the slow start threshold gets diminished unnecessarily.



(a) Hidden node problem



(b) Exposed node problem

Figure 2-4. MAC protocol problems in TCP point of view

[1] G. Xylomenos et. al. [156] argued the *Self Collision* problem because of the half duplex link of 802.11 and H. Wu et. al. [152] proposed a modification to the DCF 802.11 MAC protocol to alleviate this self collision problem.

As TCP end-to-end approaches, [153, 155] evaluated that using smaller values of both packet size and maximum window size in TCP configuration can alleviate such problems to some extent in the sense that the less end-to-end traffic being injected into the network (the less medium contention), as well as the smaller packet size, the less time taken for packet forwarding. The number of collisions is then reduced, and local MAC retransmissions are more likely to succeed by multiple attempts. Moreover, [154] has shown that the delayed ACK option can also improve TCP throughput because the number of ACKs will be reduced by half, being propagated in every two RTTs.

2.3.4 Evaluation of flaws of 802.11 random medium access

Nodes involved in ad hoc forwarding are identified by specific ranges of valid transmission and interference as shown in Figure 2-5.

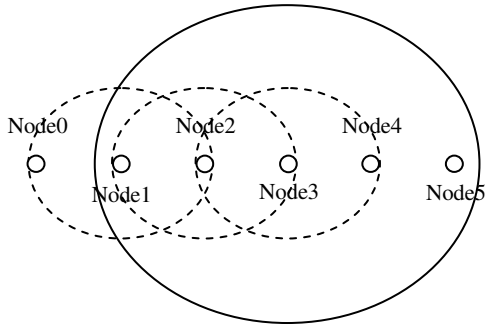


Figure 2-5. Interference range affecting the Ad hoc forwarding (the solid line denotes the interference range of a node and the dotted-line the valid transmission range)

The simulator used is the ns simulator in support of CMU wireless extensions, and the parameters of 802.11 MAC are tuned to model the 914MHz Lucent WaveLAN DSSS radio interface card at a 2 Mbps data rate. In the simulator, the effective transmission range is 250 metres and the interfering range is about 550 metres as shown in Figure 2-5. Note that one node can interfere with the packet reception of another node even though nodes are too far apart for successful transmission (being separated by 200 metres just under the transmission range is likely to achieve close to the maximum capacity possible in the sense that the higher node density, the less capacity a node occupies among the competing nodes).

In terms of achieving the spatial channel reuse to give effective bandwidth utilization with regards to interdependency of medium availability, only other nodes at least 3 hops apart from the currently communicating pair are able to transmit concurrently without any MAC collision. For instance,

once communicating between node 0 and 1, only node 4 and 5 can communicate with each other at the same time because, if other intermediates (2 and 3) make transmission, it might interrupt node 1's reception. The 802.11 RTS/CTS mechanism ensures no transmission is made by node 2 and 3 because node 0 and 1 and node 4 and 5 must put them into the NAV state eventually which ends up with to the spatial channel reuse; however, due to an improper set of backoff intervals in practice, both of the transmissions made between node 0 and 1 and node 4 and 5 might be not likely to be transmitted over at the same time even though both nodes have packets to transmit, and thus an optimum timing scheme may be required in a global view of the network.

Looking at the functionality of the NAV in 802.11, when node 3 is transmitting to node 4, node 1 does not initiate RTS propagation due to the distant interference from node 3, because node 3 has put node 1 into the NAV state, even though communication from node 1 to 0 is likely to succeed because node 3 does not interfere with both the origination and reception of node 0. In that case, node 3 may prevent node 1's transmission almost always putting node 1 into the NAV until node 3's communication ends (i.e., well known exposed terminal problem) and thus ends up with to throughput degradation (e.g., in the TCP aspect, stalling the reverse ACK flow may cause spurious RTO at the sender). In the other round, in the case of node 0 transmitting to node 1, node 3 interferes with the reception of RTS or data frame at node 1 and, when heavy traffic interferes, completely blocks. Consequently, after MAC retry expiration, the link between node 0 and 1 will be considered as temporarily broken and then drop all packets queued.

Interestingly, it can be seen that burst traffic flows between node 0 and 1 can also stall the flow between node 3 and 4 putting node 3 into the NAV state. It means for example that when a TCP connection is made from node 0 to 4, 2-hop preceding packets residing at node 3 may experience delays due to medium occupation by, in this case, subsequently traversing packets between node 0 and 1. Therefore, due to the interdependency of nodes to access the medium, high contention at a certain point along the path might be likely to be perceived by (1- and 2-hop) preceding packets that give the TCP receiver a highly fluctuating forward link delay prior to subsequent packets that actually competed from the contention. In other words, medium contention at a certain hop must have a wide effect on determining path condition, resulting in (likely) highly fluctuating forward link delays of packets.

```
s 10.561125033 _3_ MAC  --- 0 ACK 38 [0 4 0 0]
s 10.561161704 _0_ MAC  --- 0 RTS 44 [73e 1 0 0]
r 10.561429700 _4_ MAC  --- 0 ACK 38 [0 4 0 0]
```

D 10.561514371 _1_ MAC COL 0 RTS 44 [73e 1 0 0]
(Collision at node 1 made by transmission of two hop away node 3)
s 10.583244319 _0_ MAC --- 0 RTS 44 [73e 1 0 0]
(RTS frame resent after a MILD backoff interval)
s 10.583279037 _3_ MAC --- 0 RTS 44 [253e 4 3 0]
D 10.719118526 _1_ MAC COL 0 RTS 44 [73e 1 0 0]
(Collisiom ocuurred again by the interference of node 3)
D 10.726027859 _0_ MAC RET 0 RTS 44 [73e 1 0 0]
(MAC retry count expired, denoted RET)
D 10.726027859 _0_ RTR CBK 304 cbr 100 [13a 1 0 800] ------- [0:0 1:0 30 1] [279] 0 0
D 10.726027859 _0_ RTR CBK 307 cbr 100 [0 1 0 800] ------- [0:0 1:0 30 1] [280] 0 0
(Drops packets in the interface queue)
D 10.726027859 _0_ MAC --- 304 cbr 100 [13a 1 0 800] ------- [0:0 1:0 30 1] [279] 0 0
(MAC drops the packet attempted)
s 10.726102859 _0_ MAC --- 0 AODV 92 [0 ffffffff 0 800] ------- [0:255 -1:255 1 0] [0x8 2 [3 1] 0.000000]
(Immediately initiate routing control packet for rerouting)

Table 2-1. ns packet trace to show frequent interruptions of reception of two hop away neighbor.

Table 2-1 verifies that when simulated to have a video-like UDP flow (i.e., 200 Kbps CBR source of packet size 512 bytes) from node 0 to 1 and a FTP flow (of packet size 500 bytes) from node 3 and 4 (in 6-node stationary sting topology under AODV routing protocol), the burst FTP initiations at node 3 interrupt the reception of RTS frames at node 1 and, even though there is successful RTS-CTS exchange, the reception of UDP data frame. So, a number of local retransmissions take place. As seen, MAC retry count had expired, and in turn node 0 started to rediscover a new route immediately. After a new route is established (i.e, routing control packets by AODV to reconfirm the route sustainability), node 0 can transmit UDP packets immediately after the RTS-CTS handshake is successful. Of course, burst UDP flow hampers the FTP flow, and vice versa; by controlling the transmission rate of the FTP flow, it could give the UDP flow more priority because FTP is a congestion-sensitive and delay-tolerant flow.

Now, suppose two TCP traffic sources are contending with each other. FTP TCP sources were applied at node 0 (sending towards node 1) as well as node 3 (sending towards node 4). By interfering with each other, TCP exponential RTO backoffs substantially degrade TCP performance. While node 3 interrupts node 1's reception, the time spent on local retransmissions due to the medium contention may exceed the RTO. Moreover, after the route is given up due to expiration of MAC retry count, TCP packets queued are all dropped.

As seen in Table 2-2, even when a route is still sustained to be able to transmit, TCP source will not produce any further packets due to still being waiting for (exponentially backed-off) RTO expiation. As thus required, when the breakage was a multi-hop away from the sender, a probing or explicit link restoration notification is strongly necessary to quickly invoke fast transmission subsequently without need of awaiting the RTO expiration. Alternatively, an inter-layer flow mechanism between layered protocols is necessary to make an immediate transmission as soon as a route is available [137].

(The link between node 0 and 1 has been broken by being apart, TCP timeouts)

s 17.000000000 _0_ RTR --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 12 [4 9] [0 32]]

s 17.000075000 _0_ MAC --- 0 AODV 100 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 12 [4 9] [0 32]]

r 17.000875667 _1_ MAC --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 12 [4 9] [0 32]]

r 17.000900667 _1_ RTR --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 12 [4 9] [0 32]]

s 17.008799193 _1_ RTR --- 0 AODV 48 [0 ffffffff 0 800] ------- [1:255 -1:255 29 0] [0x2 2 12 [4 9] [0 32]]

s 17.008874193 _1_ MAC --- 0 AODV 100 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 12 [4 9] [0 32]]

r 17.009674859 _0_ MAC --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 12 [4 9] [0 32]]

r 17.009699859 _0_ RTR --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 12 [4 9] [0 32]]

(In turn, routing protocol keeps finding a new route to forward)

s 17.869248577 _0_ AGT --- 453 tcp 552 [0 0 0 0] ------- [0:0 4:0 32 0] [207 0] 0 0

(In the meantime, TCP agent in the slow start phase resends a packet with a doubled RTO, 8 secs)

r 17.869248577 _0_ RTR --- 453 tcp 552 [0 0 0 0] ------- [0:0 4:0 32 0] [207 0] 0 0

(TCP agent passes it down to the routing agent)

D 19.500000000 _0_ RTR NRTE 451 tcp 572 [0 0 0 0] ------- [0:0 4:0 30 0] [207 0] 0 0

D 19.500000000 _0_ RTR NRTE 452 tcp 572 [0 0 0 0] ------- [0:0 4:0 30 0] [207 0] 0 0

D 19.500000000 _0_ RTR NRTE 453 tcp 572 [0 0 0 0] ------- [0:0 4:0 30 0] [207 0] 0 0

(Routing protocol gives up finding a route, drops packet queued.
Notice that three packets whose sequence number is 207 had been passed through
the TCP agent. Each packet has a doubled RTO)

s 25.869248577 _0_ AGT --- 454 tcp 552 [0 0 0 0] ------- [0:0 4:0 32 0] [207 0] 0 0

(After 8 seconds, TCP resends a packet with the doubled RTO of its previous, 16 secs)

r 25.869248577 _0_ RTR --- 454 tcp 552 [0 0 0 0] ------- [0:0 4:0 32 0] [207 0] 0 0

(Routing agent reinitiates the route rediscovery procedure)

s 29.500000000 _0_ RTR --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 13 [4 9] [0 34]]

s 29.500075000 _0_ MAC --- 0 AODV 100 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 13 [4 9] [0 34]]

r 29.500875667 _1_ MAC --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 13 [4 9] [0 34]]

Table 2-2. ns packet trace to infer the problematic RTO exponential backoff in case of route disconnection. It shows that TCP agent will not pass down a further packet even though a route has been found.

As a proactive remedy against the considerable medium contention problem in high contending case, the TCP sender end might throttle its transmission window in advance and so can alleviate the contention problem to an extent because MAC layer retransmissions (followed by the substantial access delay of a binary or MILD contention window backoff mechanism) will impose corresponding end-to-end delay (i.e., it is likely to cause spurious RTO expirations). Thus instant route breakages occurred by the expiration of MAC retry count could be avoided by means of a timely throttling of transmission rate (because lowering sender's rate can mitigate medium contention degree widely applied to the path link).

Though the deficiencies have been recognized, 802.11 is still challenging for ad hoc networks because of its simplicity of use and over a single common channel provides promising performance reasonably well among a number of competing nodes. The determination of the common medium availability by a given end node is a critical key point because the medium is in use widely throughout the network. In this sense, this thesis endeavors to try to reasonably vary the TCP sender's transmission rate according to the medium contention, whose extent is assessed by the TCP receiver (because the TCP receiver recognizes in advance to the sender TCP).

2.3.4.1 802.11 in practice

In reality, nodes are unevenly charged and thus have different transmission and interfering ranges. In turn, although the MAC layer backoff scheme plays in lessening MAC collisions, they suffer from impaired medium reservation capability of RTS-CTS handshake resulting in unnecessary backoffs of the contention window. N. Poojary [120] noted that heterogeneously battery-charged nodes behave worse than identically charged nodes. Thus for better forwarding capability, nodes should be equally charged to have identical transmission and interfering ranges. So, the proposed mechanism is able to extend impaired transmission range due to less charged battery to the identical transmission range by relaying RTS/CTS frames. Apparently, it could improve the forwarding ability and aggregate throughput substantially. However, it failed to evaluate its outperforming due to the impact of additionally high MAC overheads required. Thus it further encourages investigating a more effective reservation mechanism.

2.4 Ad hoc TCP

Transmission Control Protocol (TCP) is so widely deployed that most end-hosts use it to provide a reliable packet delivery. It is defined basically by RFC 793 [122], RFC 1122 [35] , RFC 2581 [11] and extended by RFC 1323 [33] , RFC 2018 [101], RFC 3168 [124]. The network end-hosts, which share a best-effort network without the notion of admission control or resource reservation to control the imposed network load, implement their own congestion control algorithm and hence ensure network stability among a variety of pairs of correspondents that simultaneously use the network resources.

TCP basically uses the window–based congestion control in a way that the sender-end, either proactively or reactively estimating the available network bandwidth, should keep the controlled window dimensioning the total number of packets that can reside within the network (*network pipeline*).

Over past decades, there are two main congestion control algorithms proposed and deployed widely in real networks, namely TCP Reno [76, 77] (as an enhancement of TCP Tahoe with the fast transmit and recovery algorithm) and Vegas [3, 4].

Even though many other TCP variants exist from the primary TCP Reno and Vegas, mainly TCP Reno, and its variant, is widely used today, employing a reactive based congestion control mechanism. TCP Vegas has a proactive-based congestion control mechanism and thus predicts when congestion is about to happen in order to control the transmission rate and, in turn, to prevent packet loss.

In the growing Internet today, disparate types of networks are interconnected for communicating with each other. The growth proposes a number of enhancements in order to alleviate the impact of encountered problems. In terms of flow control, end hosts employ more intelligent TCP schemes, with regard to wireless error, congestion loss, or routing error. In the meantime, network routers become more elaborate to effectively control passing-through flows, such as, RED, and QoS RIO [73] (RED with IN/OUT) algorithm of the proactive-based Active Queue Management, RFC 2309 [34]. Moreover reliable link layer protocols are also deployed to improve the reliability of local hop link; in total, protocol enhancements made at one layer may affect other layer's performance and it becomes more complicated to evaluate the performance gain of a specific enhancement made.

2.4.1 Basic TCP functionality

With bandwidth provided by lower layer protocols, the sender initiates packet transmission. Each time a packet is transmitted and then succeeded by reception of ACK, the sender can increase the congestion window in a sliding window manner. Each ACK received advances the left edge of the sliding window to allow more data to be sent as long as the receivers' window (*adwin*) allows. If no packet loss occurs, it evolves the congestion window exponentially per RTT up to the slow start threshold (*ssthresh*), meaning that, each time an ACK arrives, it enlarges the congestion window by one segment up to *ssthresh* (depending on whether delayed ACKs are in use). Afterwards, a specific congestion avoidance algorithm is invoked to probe spare bandwidth. In a loss-based TCP Reno, or its variants, each ACK reception increments the congestion window (*cwnd*) by $MSS^2 / cwnd$ in byte (i.e. one MSS per RTT[1]). In delay-based TCP Vegas and its variants, the bandwidth-delay product is a key metric to adjust the congestion window in order to prevent packet loss; however, TCP Vegas cannot be directly applied to the ad hoc network because, due to dynamic node mobility and, in turn, frequent route failures, the TCP sender suffers from inconsistency of path link RTT. More details related to the TCP congestion issues are found in [11, 135, 136].

If the sender encounters a packet loss on the way, it quenches its congestion window in two different ways according to the type of perception of the packet loss (i.e., timeouts and duplicate ACKs). On timeouts, the sender shrinks the current transmit window and restarts from a small value, typically one or two segments as the initial window [11], and then progresses as supposed. In case of fast retransmit, the sender typically requests three duplicate ACKs to immediately resend the packet presumably being lost (as perceived light-weight congestion). In turn, it quenches the current transmission window (i.e., max (*adwin, cwnd*)) by a factor of 2 to set *ssthresh* and then restarts from $ssthresh + 3 \times MSS$ with increment of one segment per further incoming duplicate ACK in the sense of the fast recovery. Meanwhile, new data may be sent if the incremented congestion window becomes greater than the number of unacknowledged segments. Some time later, if the sender receives a new ACK acknowledging the packet resent, it escapes from the fast retransmit and recovery phase, and is put into the congestion avoidance phase with the *ssthresh* that was set before.

As a modified TCP Reno for error-prone wireless links, TCP NewReno [61] enhanced the fast recovery mechanism to recover packet losses (at least) one packet per RTT in support of the *partial* acknowledgement. If a new ACK, received for the lost packet resent, did not acknowledge the next

[1] $1/cwnd$ per RTT if *cwnd* is expressed by in multiple units of MSS, not bytes.

new packet but an outstanding in-flight packet of the previous window, it supposes another packet was lost in the window and immediately resent, staying in the fast recovery phase because it supposes an additional packet loss was perceived. That is, the sender can recognize a packet loss every RTT without awaiting RTO expiration. When a new ACK is acknowledged for the next window, it will get out of the fast recovery phase. On the other hand, TCP with SACK option, RFC 2018 [101], was devised to recover more than one packet loss per RTT. Performance comparisons among Tahoe, Reno, and SACK are found in [53], and among Tahoe, Reno, NewReno, and SACK are found in [138] specifically for the ad hoc network.

2.4.1.1 Drawbacks and key requirements

As briefly pointed out in early Chapter, when normal TCPs are deployed in ad hoc networks, it is definite that each sender wrongly infers the cause of packet loss because three factors, high BER, medium contention, and route disconnection, can cause a packet loss. That is, it is impossible for the normal TCP sender to classify the cause of packet loss by the traditional means of RTO and duplicate ACK.

Therefore, our ad hoc TCP enhancements do not try to identify the cause of packet loss but adjust *ssthresh* more dynamically and meaningfully for the network characteristics (rather that just shrinking congestion window drastically). More specifically, the receiver will inform the sender of an estimated *ssthresh*, which indicates the bandwidth offered by lower layer protocols, and accordingly the sender will contorol the congestion window, as well as its *ssthresh*, when it encounters timeout or fast retransmit or some other times necessary.

2.4.2 The origin of packet loss

The following introduces major reasons to cause packet losses over ad hoc links. It appears necessary enhancements in both ends in the transport layer perspective.

2.4.2.1 Path disconnection

Concerning high mobility applied, on-demand routing protocols, such as DSR and AODV, interact with MAC protocol (i.e., MAC promiscuous mode) as well as use an efficient connectivity management strategy so that it is able to determine link connectivity before an actual transmission is

initiated. A particular use of connectivity management provides better results for the sake of saving bandwidth as far as efficiency and delays are concerned. For example, AODV [52, 72, 117] has several connectivity support ways (e.g., incorporation of 802.11 MAC protocol such as the absence of link layer ACK or failure to get a CTS after sending RTS even after the maximum number of retransmission attempts, and MAC eavesdropping, Hello messages using either RREQ or RREP, and a combination of MAC and Hello with RREQ); periodic broadcasts of connectivity-concerned messages might have negative effects on network congestion and battery lifetime.

If no specific connectivity management in a routing protocol exists, the detection of link breakage is recognized after a certain number of successive medium acess failures. Typically, after a certain number of unsuccessful attempts of RTS-CTS frame exchanges in the MAC layer, the frame initiator gives up the transmission of the packet, drops it and then concludes that the next hop link was broken. The information of the hop link breakage is passed up to the routing protocol, and consequently the routing protocol gives a feedback to inform the sender of the route error in order to invoke a route rediscovery procedure. Such local MAC layer attempts may however interrupt the reception of other in-flight packets and block subsequent packets that were being queued but travel through a different link still sustained because of a single interface queue in the mobile node. Thus, in this case, relying upon the MAC layer-based detection underutilizes the available network resources among multiple flows because it cannot distinguish actual link breakage (due to, say, partitioning) from high medium contention-induced instant link breakage. On the other hand, in case of highly moving situation, route disconnection may be more likely to occur due to being out of coverage of transmission range rather than due to high medium contention; therefore, an explicit signaling to check the link connectivity should be required for avoiding unnecessary data transmission.

2.4.2.2 Wireless link corruption

In wireless links, in terms of designing a more delicate TCP scheme to identify the cause of packet loss correctly, various strategies have been proposed—Balakrishnan et al. [18], Bansal et al. [23], Goel et al. [67], Fei et al. [55], Samaraweera [130], implemented schemes where the TCP sender was explicitly informed or implicitly calculated by itself whether a packet loss was due to congestion or wireless link error. Basically, the properties of wireless links are very different from those of wireline links: Wireless links are characterized by high BER with random losses caused by shadowing and fading.

For instance, with a random link error in a wireless network, TCP misinterprets the packet losses as congestion related losses; consequently, a single error caused by the link will lead to duplicate ACKs and TCP will invoke the usual congestion control mechanisms. The fast retransmit and recovery scheme in general[1] will eventually decrease the transmission rate by a factor of 2 even if congestion is not present in the network. Such spurious decrease will underutilize the available bandwidth. Two studies [18, 130] suggest that over wireless links, maintaining the congestion window will further improve the throughput, being coupled with SACK option

Furthermore, the link may cause burst errors when the link is in a deep fade for a significant amount of time, and in cases where the deep fade may span more than one window and cause packet losses across more than one window. During the long fade period, it is more effective to drop the transmission rate even if no congestion is present. Thereby, care should be taken to give the sender an ability to decide reasonably whether packet losses were due to instant link corruptions or the deep fades that the network may inform the sender, for example, by means of EWLN (Explicit wireless loss notification) [23] or WECN (Wireless ECN) [55]. The disadvantages of these proposals as expected are that the EWLN notification needs the base station to keep track of all the packets going through it requiring more intelligence and that the WECN requires changing the code at the routers as well as at the senders and is thus impractical to configure the entire set of transmitting nodes if they belong to heterogeneous networks.

2.4.2.2.1 Reliable link layer scheme in use

In addition to the link layer Automatic Repeat reQuest (ARQ) protocol, Forward Error Correction (FEC) coding scheme [40], or other link layer schemes [18, 113], 802.11 has the link-level acknowledgment of data transmissions so that every data transmission will be followed by link level ACK from the MAC-level receiver for the sake of integrated hop-by-hop reliability. Whereas, other primitive MAC schemes such as CSMA and MACA do not have the link-level acknowledgment, so have to make retransmission initiated by the transport layer, resulting in significant delays.

Specifically, 802.11 typically attempts 7 times to ensure the successful RTS-CTS frame exchanges and 4 times to ensure the successful data frame reception, which could be enough to recover

[1] In particular, TCP Westwood [99] introduces, namely, faster recovery mechanism for which in case of either timeout or fast retransmit triggered, the sender selects a slow start threshold and a congestion window that account for the effective bandwidth estimated.

wireless hop-link corruption[1] made between nodes even though the additional attempts cause substantial delay; subsequent or preceding in-flight packet(s) in the network might be delayed accordingly. In the meantime, RTT is being inflated and fluctuated, packet by packet, by the link level effort—that is, the expense of the link level ACK scheme mitigates the impact of relatively higher BER, and the resultant delay contributed by the high BER will be comprehensively included in the term, *comprehensive MAC related delay* (Chapter 3.4.2.1), by which the receiver throttles the sender's transmission window.

As a consequence, we can conclude that 802.11 imposes considerable MAC related delays when either higher BER or heavy contention is applied and so, using that, the receiver can recognize the delay and probably throttle the sender's transmission window at the instant. As will be proposed, the receiver will keep track of forward link delay (FLD) and its deviation (forward link delay deviation, namely FLDD), and, if higher BER as well as medium contention induced is applied occasionally and the receiver perceives considerable FLD and FLDD, the receiver will reduce the sender's transmission rate by a feedback, and sometimes even prepare to freeze the sender to prevent further transmissions, to cancel RTO timers, and to renew the current *ssthresh*.

2.4.3 The need of a newly designed TCP scheme

In the wireline networks, the bottleneck routers occasionally suffer from queue overflowing problems because many traffic sources are using the router as a common transit and so the incoming traffic amount may overwhelm the outgoing traffic capacity. Many proposals dealing with rate control mechanism, originated by the bottleneck router, have been addressed to estimate a suitable explicit rate of individual end host by monitoring the outgoing buffer level of the router (i.e., ATM's ABR rate based congestion control as a general framework for feedback based explicit bandwidth signaling [32, 41, 144]).

As said previously, however, in wireless networks, particularly in ad hoc environments, the network load capacity is mainly signified by wireless link contention. As clarified in [63], packet drops by buffer overflow were never observed but all packet drops were due to the medium contention, given that each router has reasonable buffer size. Thereby, the reception of ACKs no longer implies the buffer condition along the path but rather the currently available medium condition. There have been numerous research papers which addressed the MAC related TCP throughput analyses [25,

[1] The presence of burst wireless channel errors will thus lengthen MAC related delay resulting from the considerable link level effort, and as a result cause additional MAC contention with other adjacent nodes.

118, 142, 143, 153], to improve either the efficiency of aggregated throughput, or fairness among flows. But, few explicit rate controls made at the end-to-end perspective exist, in order to handle the wireless medium among many competitors. This thesis is thus primarily motivated by the need of a reliable TCP scheme that is supposed, for the ad hoc receiver's enhancement (Chapter 3.3), to resolve congestive medium contention to a considerable extent (comprehensively including the impact of higher BER) and, for the ad hoc sender's enhancement (Chapter 3.5), to protect against vulnerable route connectivity during the connection time. In order to resolve the former problem, the receiver should collect the information about the current network condition. The following section lists some of typical paradigms where the end node can be aware of the network information, and addresses the shortage of schemes which rely upon network-originated feedback.

2.4.3.1 Justification of inter-layer operations to give flow control

The dynamic nature of network topology over a single common channel to be shared requires one to understand the complex interactions between TCP and lower layer protocols. As a matter of fact, vertical information flow required between layers violates the standalone modularity of TCP stock of a conventional flow that only uses strict peer-to-peer horizontal communication between layered protocols; however, flexible inter-layer information flow with lower layer protocols can provide a useful knowledge of network link condition because routing protocol plays a role in sustaining path link between end TCPs and because MAC protocol is responsible for per-hop basis bandwidth provision.

D. Sun and H. Man [137] proposed for ad hoc networks the enhanced inter-layer control mechanism (ENIC) so that it enables layer-to-layer vertical communications among layered protocols employed, where upper-layer protocols bind more closely with lower-layers leading to more efficient and direct inter-layer operations that drastically reduce the overhead of layer-specific control messages.

Tom Goff et al [68] introduced the use of freeze TCP for handoffs to necessitate freezing in response to impending link disconnection determined by lower layer protocol at routers on the way. The freeze TCP does not require modification of the sender but requires the prediction of impending disconnection related to the hop link condition determined by MAC protocol (namely TCP aware link layer scheme). We thus believe that an interoperating mechanism between layered protocols justifies determining such dynamic bandwidth availability.

The following paradigms are network feedback based approaches for reactively handling disconnections over multihop ad hoc links. TCP-ELFN [74, 104] uses ELFN (Explicit Link Failure Notification) occasionally originated by a router whose next hop link was broken, so that the TCP end host can take the information and stall transmission until the link gets restored. As a drawback, the sender and the receiver rely upon the performance of all intermediate routers along the path, where routers propagate the ELFNs according to the link viability. Also the sender must be modified to function with the ELFN message.

ATCP [91] also uses two kinds of explicit control beacons, such as ECN (Explicit Congestion Notification) and ICMP, in order to control flow rate (ECN message determines when the network is congested, distinguishing from instant wireless link corruption, and ICMP "Destination Unreachable" message informs of whether the network is partitioned, or when no route exists). In fact, the sender's behavior in terms of flow rate control mechanism should not be fully dependent upon the receipt of any of two because ICMP message is transmitted via the UDP protocol that does not guarantee its delivery and because ECN-designated packet may also be lost over links. For example, the loss of the ICMP message results in a number of retransmit timeouts followed by a number of retransmissions successively since the disconnection period would have taken place. Besides, ECN cannot be relied on to completely eliminate packet losses as indications of congestion and therefore would not allow the end nodes to interpret packet losses as indications of wireless medium losses (corruption due to high BER) instead of congestion. In addition, all the routers along the path link must be ECN-enabled.

In fact, the explicit network feedback is more accurate and guarantees timely response, but requires considerable network bandwidth consumption by intermediate routers. So, it might not want to spend this if there is not reasonable performance improvement by use of it. As a result, the most preferred strategy will be any of those that do not fully rely upon the network oriented feedback, even though it can be improvable, but also upon the end-to-end TCP scheme because it can facilitate useful information available from the lower layers by the inter-layer vertical communication.

2.5 The State-of-Art existing TCP schemes

Many proposed TCP schemes may be classified as network detection- and end node detection-based approaches. For example, TCP-Feedback [39], ELFN-based [74], ATCP [91], TCP-Bus [86], TCP-

EXACT [42], and ATP [139] approaches rely upon network signaling for detecting path anomalies. So, they are related to network detection strategy and dominated by the fidelity of the network based detection mechanism. This means the reliability of the detection strongly influences the end-to-end flow control. In particular, TCP-EXACT [41] and ATP [139] rely on the network originated information to control the sender's flow rate and thus its accuracy matters.

On the other side, TCP-DOOR [56], Fixed RTO [49], ADTCP-friendly [58], and Edge-based TCP [110] approaches are related to end node detection without support of any intermediates in the network, so keep the end-to-end TCP semantics in flow control.

Each approach has considerably advantages and disadvantages in some aspects, and so the ideal should be a hybrid approach based on a proper tradeoff to complement each other. In the sense of that, the network detection approach is able to inform the end nodes of the network condition more accurately and quickly since the intermediate nodes detect failures more quickly than the end nodes would do by using the end node detection scheme. The end node detection, however, has an advantage that it does not need intermediate nodes cooperation, which is highly desirable for the sake of overhead, security and so on.

For a typical example of the end-to-end approach in terms of frequent route change problem, F. Wang and Y. Zhang [56] introduced purely end-to-end based TCP-DOOR so that either sender or receiver works without aid of an explicit network based control beacon in order to detect whether the route has been changed (detailed in Chapter 2.5.5).

G. Ahn et. al. [2] introduced rate and admission controllers, namely SWAN, functioning at sender-end monitoring incoming packet delays, taking into account traffic types (i.e., best effort traffic and real-time traffic), and performing without support of any network feedbacks. The SWAN AIMD rate control mechanism is capable of maintaining the MAC delay under a certain target boundary. However, it does not provide a complete TCP flow control scheme.

ADTCP-Friendly cited in [58], likewise, uses end node determination so that the receiver end categorizes the network condition with several metrics, IDD (Inter-packet delay difference), STT (Short-term throughput), PLR (Packet loss rate), and POR (Packet out-or-order delivery rate), then makes a decision through two kinds of decision inference algorithms (i.e., Threshold-based and Maximum likelihood classifiers) with inputs of the metrics measured for cases of packet absence in

sequential arrivals, at which the receiver is able to determine whether the network link has got to any of anomalies categorized, such as congestion, router change, channel error, or route disconnection stage. In response, the sender will take a reasonable action accordingly. However, these end-to-end approaches do not propose an explicit rate control in the window progression of the sender but give a control by inferring the cause of packet loss only in case of out-of-sequential packet delivery.

As a major drawback, however, both approaches are likely to require specific-purpose modifications in either end, or both, or intermediates and in turn might not guarantee backward compatibility and its transparent, robust functionality when interoperated with an unmodified counterpart in a heterogeneous network like the wireline Internet and could eventually end up with sub-optimal behavior due to the incompatibility.

2.5.1 TCP-Feedback [39]

As an early heuristic strategy to propose to eliminate unnecessary RTOs, TCP-Feedback (TCP-F) [39] introduced the feedback based scheme where the source can distinguish between route failure and network congestion by means of a RFN (Route Failure Notification) when the route is disrupted, and RRN (Route Re-establishment Notification) when the route is re-established. As drawbacks, no consideration was made for congestion window size to be adjusted whenever a new route has been discovered, and in practice, there is no such routing protocol facilitating such control messages, RFN and RRN.

2.5.2 ELFN-based [74]

As an early proposal promising against the nature of frequent link breakages, G. Holland et al. advocate the use of the ELFN to improve the end TCP performance. Simply speaking, each time a router along the path encounters the next hop link broken, the TCP sender is informed of the link failure by means of the ELFN originated by the router that detected the failure. In reception of the ELFN, timers and window size are frozen as in TCP-F. At a regular interval, the sender probes to know if a new route is available.

On the other hand, J. P. Monks and et.al. [104] presented the limitations of the ELFN signaling for use in static, as well as dynamic, networks, as well as demonstrated limitations of TCP based

congestion control mechanisms for ad hoc networks, leading to conclude that hop-by-hop rate control mechanisms along with ELFN are better suited.

2.5.3 ATCP [91]

ATCP proposed a thin layer, so called 'ATCP', inserted between IP and TCP. It listens to network state information such as ECN and ICMP. This layer is responsible for putting TCP into the appropriate state depending on the network layer feedback, which places TCP into one of persist state, congestion control state or retransmit state, relying on ECN (Explicit Congestion Notification) messages to enable it to determine when the network is congested and on ICMP "Destination Unreachable" messages to inform it whether the network is partitioned or when no route exists.

This scheme conserves end-to-end TCP semantics and does not interfere with TCP's congestion control behavior when the network is congested. As a drawback, the functionality of ATCP fully relies on ECN to invoke congestion control and ICMP message to inform of disconnection. This ICMP message originated from the network can better handle the event of unpredictable route disconnections but might cause the network link to deteriorate due to considerable bandwidth expenses.

2.5.4 TCP-Bus [86]

D. Kim et. al. introduced TCP-bus as an intelligent flow control algorithm particularly for cases of frequent route disconnection and rerouting. It takes the awareness of packet sequence numbers, the packets of which were generated from the sender, some of which resided in the network as in-flight packets outstanding, and the other of which was successfully received from the receiver, in order to avoid unnecessary retransmissions at times of route failures explicitly notified.

This protocol is implemented in support of dedicating ABR routing protocol. ABR is modified to generate two control messages such as ERDN (Explicit Route Disconnection Notification) and ERSN (Explicit Route Successful Notification), whose parameters are ERDN_GEN_SEQ (as the sequence number of the TCP segment pending in the head of line of the intermediate node's transmit queue) and Last_ACK (as the sequence number of the last segment until the destination has received it successfully), respectively, in order to handle the route failure cases properly at the source TCP.

Once the route has been disconnected, an intermediate node, namely *pivot* node, that detected the disconnection will inform the sender of two sequence numbers, via ERDN (Explicit Route Disconnection Notification) signal, one of which is the sequence number from which it will drain out after route reestablishment (i.e. ERDN_GEN_SEQ) and the other of which is the last sequence number buffered successfully from the sender (i.e. ERDN_RCV_SEQ).

During the route discovery, it may heap in-flight packets in its buffer waiting for forwarding after the route reestablishment. When the route has been reconnected, the intermediate will send an ERSN (Explicit Route Successful Notification) signal back to the sender to inform of the last sequence number successfully received at the destination.

In brief, each time the route is disconnected, the pivot node should deliver the ERDN message confidently to the sender because the sender has to take the awareness of what packets are buffered waiting for reconnection at the intermediate, so that the sender can skip those packets and so will not unnecessarily retransmit.

To guarantee the delivery of the ERDN message, it has utilized ERDN_RET_TIMER so that it will retransmit the ERDN message unless it cannot overhear the ERDN message relayed by the next neighbor node within the ERDN_RET_TIMER period. In addition, in order to be convinced of the delivery of ERSN from the intermediate to the sender, the sender starts to probe to check route reconnection periodically.

Drawbacks include: TCP-bus is too susceptible to the control beacons of routing protocol and also relies upon the performance of MAC protocol for eavesdropping. Also, there is considerable computational effort required at the pivot node if there are many competing nodes using a common router. In addition, there exist a large number of out-of-order packets at the destination due to reroutings, and the functionality of intermediate nodes does not work properly in use of IPsec.

2.5.5 TCP-DOOR [56]

The study proposed in [56] addressed a simple implementation called TCP-DOOR (Detection of Out-of-Order and Response) that does not use any feedback of either network-originated information or lower layers but yields on an average 50% performance improvement.

Out-of-order packet delivery is an important way to distinguish route changes from network congestion, detected either at sender or at receiver. Hence, out-of-order delivery of packets at sender/receiver means that, the route between corresponding nodes has been changed and the sender should not invoke congestion control algorithm. However, if it is complied with a routing protocol that sometimes prefers multipath routing and so results in out-of-order delivery, TCP-DOOR will respond wrongly.

This paper however implies that frequent out-of order deliveries of packets (i.e data or ACK) at sender/receiver end would give a meaningful notification of network path switches, of course, even though it is dependent upon routing preferences.

2.5.6 Edge-based TCP [110]

The paper has described heuristically an interesting insight into using RTT variations for discriminating packet losses due to medium error and disconnection from congestion induced losses and then has shown the likelihood of RTT measurement-based inference to accomplish an end-to-end TCP scheme independent of intermediate cooperation because RTT variations may well reflect congestion state inside the network.

As verified by its ability of discrimination, the RTT variations can reveal that it could indicate network congestion accurately to an extent but only in certain particular cases. For example, under heavy congestion, RTT variations proved not to indicate much about the changes inside the network. However, as time evolves, useful information could be still extracted from the RTT behavior being put to further research; as a similar context, our proposition in use of FLD and FLDD further justifies the useful information extraction from the packet traveling time.

However, in terms of the analysis of the RTT variation, [123] has argued that simple RTT variation schemes can not always predict congestion from wireless link error well, and moreover they do not work well under all kinds of simulation topologies and traffic load conditions.

2.5.7 ATP [139]

The study cited in [110] introduced an entirely new TCP scheme for ad hoc networks. ATP requires all involved nodes to be modified. The main rationale is that in the intermediate routers, they

monitor two metrics (average queuing delay[1] and transmission delay) experienced by packets at an instant traversing through them. The two metrics are designated in the rate feedback field (D)[2] of traversing packets. In the receiver end, the maximum average delay (avg(D)), which implies a bottleneck router's bandwidth availability, will be advertised[3] by a rate feedback in a certain interval. In the sender in response to the rate feedback, it computes the next sending rate (1/avg(D)) to adjust or maintain; our enhancement is to estimate the term, the per-node average delay, at the receiver end (Chapter 3.4).

Because the feedbacks of the delay factors experienced by outgoing packets passing through each intermediate (i.e, bottleneck router) are sent back to the sources of the corresponding flows, the sources will respond in an identical manner to compute the available bandwidth. It yields a high degree of global fairness in the network.

Moreover, due to frequent path failures and resultant timeouts, the TCP sender spends considerable time in slow start phases and thus suffers from the underutilization of network resources. Beneficially with the rate computation taking a single round-trip time, during connection initiation, or when recovering from a timeout, ATP proposes a probing mechanism called the *quick-start* for which the sender is able to make a probe to check the available bandwidth and subsequently converge on the available bandwidth.

As drawbacks in ATP, the protocol is not compatible to comply with other TCP ends that do not know its modification. Intermediate routers are involved to affect TCP flow rate giving receivers the per-node delay information.

2.5.8 TCP-EXACT [42]

Likewise with ATP, each bottleneck router explicitly gives senders appropriate rates which are designated in the ER (Explicit Rate) field of each traversing packet. Each router computes the allowed rate of each flow out of all competing flows present in the router, which is based on the measured effective bandwidth representing outgoing link capacity—average throughput measured

[1] The amount of queuing delay is equivalent with the amount of MAC related delays of other preceding packets being dequeued in advance.
[2] More delay (i.e., the sum of weighted moving averages of queuing and transmission delays) is overwritten through bottleneck routers, which would then represent the available link rate.
[3] Likelihood of use of receiver's window advertisement using the maximum average delay, but it elaborated ATP sender to compute an available rate with the delay provided by the receiver.

by successful transmissions of data packets implies the normalized available bandwidth of a standard packet size, which then represents the available bandwidth of a wireless link at that instant (dominated by medium contention). Adopting max-min fairness criterion, flows with minimum requests are granted first, and then leftover bandwidth is shared equally among higher demanding flows, which guarantees fairness among competing flows.

For rate designation, packets have a flow control header that consists of two fields, ER and CR (Current rate). Initially, the sender sets the ER field as its maximum request rate for use after one RTT, and through traversing each routers along the path the ER field might be reduced by bottleneck routers (if any) to signify another less allowed data rate. The CR field means the current sending rate of the flow, which is maintained in each router, in order to compute the fair share of bandwidth among active flows passing through the router.

As verified throughout a series of simulations, TCP-EXACT outperforms Reno and SACK, and could combine SACK for reliability control and a *safety window*[1] to guard against feedback packet loss

As drawbacks, the explicit rate computation of each flow made at each router requires identifying active flows [2] at the router in order to fairly share the available bandwidth. Thus, if IPsec option may be employed, there is no way to identify flows because traversing TCP segments need be snooped in each router.

2.6 Other end-to-end schemes

In relation to bandwidth-constraint ad hoc links, any of following, or others, might be carefully considered being deployed for a better performance improvement in terms of bandwidth-limited and power-constraint properties of the mobile node. The use of any of the following is however out of the scope of the thesis and thus not evaluated.

[1] The safety window limits the amount of damage from the rate that the sender could spuriously cause due to feedback packet loss, with the upper-bound of the bandwidth delay product as size of the safety window of the minimum allowed data rate.

[2] Each router maintains a flow table with the following fields, (src_ip. src_port, dest_ip, dest_port, next_hop, refreshed_time, current_rate). On receiving a data packet, each router updates the flow's next_hop, refreshed_time (a flow need refresh itself within a certain period of time), and current_rate.

2.6.1 TCP SACK

There is a high likelihood of multiple losses in a given window in wireless links. The selective acknowledgement option (SACK) proposed in RFC 2018 [101] can allow the TCP sender to correct the multiple packet errors in a single window more quickly. The sender can infer the sequence numbers of more than one packet loss when a selective acknowledgment is received. This makes it retransmit the dropped packets in a shorter period of time. If there is no selective acknowledgement, the TCP sender should have to wait for duplicate ACKs for all the dropped packets, resulting in a longer delay. This is why the TCP with SACK option should be a more preferable TCP when applied to the wireless links.

New Reno TCP that recovers at most one packet loss every one RTT, mitigates, to some extent, the impact of multiple losses within a window. When a lost packet has been retransmitted by fast retransmit and afterwards does not acknowledge highest sequence number outstanding within the window (i.e., *partial* acknowledgment), the sender concludes another packet has been lost and immediately retransmits the packet. The sender keeps doing that until all the packets within the window have been acknowledged.

As verified in [53], SACK TCP outperforms TCP new Reno in case of multiple losses in the sense that the new Reno TCP is able to transmit at most one packet per RTT. The study has pointed out the fundamental restrictions penalized by the absence of selective acknowledgments in TCP and apparently implies that the use of SACK option should be beneficial for ad hoc networks because there are likely to be multiple losses for short time scales with a small window of data. In turn, we will use TCP SACK modeled as in RFC 2018 (i.e., “sack1” in ns2) as the baseline.

2.6.2 Timestamp option

The Timestamp option in RFC 1323 [33] was introduced for window scale option for the high capacity network of large bandwidth-delay products and protection against wrapped sequence numbers for high speed network and a robust RTT measurement against retransmission ambiguity problems.

It requires the sender to timestamp every transmitted packet and the receiver to echo every arrived packet. With timestamps for which each packet additionally uses 12 bytes, every packet can be used

as an RTT sample so that it could provide a better RTT estimate in respect to spurious RTOs because timing every packet presumably much more closely track changes in RTT. As argued in [95] in the sense that in a wireless network where link characteristics can change considerably over short time scales, it is important to track the RTT as frequently as possible.

Many algorithms (e.g., [58, 112]) improving TCP over wireless links are dependent upon the timestamps. Likewise, in this thesis, the timestamp option is in use to facilitate a number of versatile information such as the following:

- The provision of a better RTT measurement in response to dynamic changes of ad hoc link.
- The primary merit in use of the time-stamped packet is that the sender can decouple either forward or reverse link delay from RTT and easily keep track of either the measurement of the instant of every packet's arrival. In this thesis, the receiver keeps measuring the forward link delay variation of data received (Chapter 3.4.4.1).

2.6.3 Eifel algorithm

In terms of detection of spurious retransmissions, the Eifel algorithm [94] uses the timestamp option employed in order to detect whether the previously retransmitted segment and subsequent window reduction were unnecessary as well as to eliminate the retransmission ambiguity problem. When the first ACK that acknowledges the retransmission arrives, the sender compares the timestamp of that ACK with the term, ts_first_rexmit (i.e., time-stamped for the retransmitted segment). If it is smaller than ts_first_rexmit, this indicates that the retransmission was spurious.

2.6.4 Transport unaware link laver improvement protocol (TULIP)

C. Parsa and J. Garcia-Luna-Aceves [113] proposed the transport unaware link laver improvement protocol (TULIP) which takes care of the out-of-order packet delivery problem, ensuring in-order deliveries to the transport layer by means of TCP unaware buffering before passing up. Thus it can improve end-to-end performance against frequent out-of-orders. It is similar to the snoop [19] protocol or delayed duplicate acknowledgements [150], but contrarily, it is not restricted to the presence of a base station and thus can easily be applied to the multihop ad hoc networks for performance improvements. However, TULIP hides the network anomalies (such as, difference of inter-packet arrival times). As a result, the transport layer is not aware of that.

2.6.5 Delayed ACK (RFC 1122 [35])

Because of the common link used for both forward and reverse links, beneficially, the use of the delayed ACK scheme [6, 154] will reduce the medium contention over the contending environment of both forward and backward links and consequently improves TCP throughput accordingly. Provided that link level ACK is in use for reliable TCP ACK deliveries, it is facilitated to minimize the traffic amount on the reverse link of ACK flow and so reduce the frequency of the sender's ACK-clocking. TCP enhanced must be able to employ the delay ACK option as evaluated in [7]. The study in the thesis does not consider and verify its gain.

2.6.6 Limited transmit (RFC 3042 [8])

Retransmit timeouts are a necessary method as a last resort in TCP flow control, which is used when the TCP sender has no other method to determine that a segment must be retransmitted. In addition, the exponential backoff of the retransmit timers is a fundamental component of TCP congestion control, particularly important when the congestion window is at most one segment. However, when the congestion window is larger than one segment, TCP can use the basic AIMD congestion control mechanisms, in the case of which it would prefer to avoid unnecessary retransmit timeouts as much as possible.
In the case of the congestion window with fewer than four segments, which is not able to receive three duplicate ACK after a loss, the TCP sender probably goes through the considerable delay of waiting for the retransmit timer to expire. It sounds unreasonable for the TCP sender to have to await a retransmit timeout to recover from the packet loss. Specifically, a relatively small window available during the whole connection time in the ad hoc network may suffer from this problem. Concerning that problem, the *limited transmit* mechanism has been proposed, and has now been approved as a Proposed Standard [8]. The sender would transmit a new segment rather than an old packet suspected to have been lost, after receiving one or two duplicate ACKs, as long as allowed by the receiver's advertised window.

The TCP sender will use this idea of the *limited transmit* that should help reduce unnecessary retransmit timeouts by invoking the fast retransmit without the wait for the timer expiration.

The *limited transmit* was implemented in the ns simulator, and in order to test this mechanism TCP connection should enable "singledup_" variable.

2.6.7 Increased Initial window (RFC 3390 [9])

When verified by a simple simulation of a stationary 5-hop string topology with a ftp bulk transfer, adjusting to an adequate, a bit increased (i.e., 1 to 4 segments), the initial window change improved TCP throughput slightly in the sense that inevitable sporadic packet losses perceived by timeout during the connection time underutilizes by a small initial window set each time timeout occurs. What that implies is that timeout does not always mean congestion but a wireless error, and thus, each time timeout occurs, shrinking to one packet is not reasonable but wastes scarce bandwidth.

As a result, it implies that according to the link condition, the sender might adjust initial window more optimally. In a similar point of view, the sender should deduce the cause of packet loss prior to shrinking to a small value each time timeout or fast retransmit occurs.

2.6.8 Dynamic RTO tuning

In RTO computation denoted in RFC 2988 [115], RTO is varied over the entire TCP session period established in response to the timings of arrival packets, historically accounting for smoothed RTT and RTT variation as in the weighted moving average. R. Ludwig [93] proposed more graceful RTO computation algorithm which can avoid spurious RTO increase due to inflated RTTVAR when a drastic drop of RTT occurs.

As a proposal for ad hoc use, each ACK arrival is timed to be sampled for measuring dynamically time-variant RTT and more closely tracking the changes in the RTT, in likelihood of changes in alpha and beta values of the moving average, which is out of the scope of the thesis, but an open research question[1], then to compute the RTO. To do so requires the incorporation of use of timestamp option that could allow the sender to sample each progressively arriving ACK for RTO computation without anxiety of retransmission ambiguousness.

[1] [10] argued that timing each segment does not lead to a better RTT estimator and further, the alpha and beta in the function of RTT computation's may keep an inadequate RTT history over time.

Besides, in [49] analyzing the performance of TCP over ad hoc networks, the authors mentioned the need of modification of the RTO (to default initial value, 6 sec, for the simulation) upon restoration of a broken route, in the common sense view that improper RTO set after the route restoration could significantly affect TCP performance because of being inaccurate with the route condition changed. D. Sun and H. Man [137] addressed for frequent changes of ad hoc links that the sender should recalculate retransmission timer after route recovery and temporarily take namely a Temporary RTO (TRTO) as a function of hop lengths in ratio of hop length of the new route and the broken route. As a result, in times of route restoration, the sender should take into account new packets, such as after a probe, to give a proper RTO value favourable to the new established route.

2.6.9 D-SACK (RFC 2883 [61])

Within the ad hoc network, the sender TCP may fast retransmit a packet after the receipt of three duplicate ACKs resulting from either actual packet loss or out-of-order packet deliveries due to route changes or link-level retransmission of corrupted packets.

For the robustness against reordering, the sender should undo the reduction of transmission rate that resulted from a fast retransmit due to three duplicate ACKs, because a number of duplicate ACKs may be invoked due to late arrivals of packets at the receiver (i.e. reordering due to route changes rather than packet loss).

Coupled with D-SACK [62], the sender is able to “undo” the previous unnecessary congestion window reduction because the sender can realize the three duplicate ACKs have been received due to reordering rather than packet loss. For instance, a routing protocol, TORA (Temporally-Ordered Routing Algorithm), which is an on-demand based protocol but has proactive features as well, was designed for high mobility and so can establish routes quickly, reducing the number of control messages in a way of confining within a small set of nodes around a link breakage. However, it does not prioritize shorter routes and might therefore yield a considerable amount of out-of-order packets and afterwards trigger unnecessary retransmissions. Therefore, DSACK undoing mechanism is useful with the reordering problem so as to minimize the impact of the unnecessary reduction of congestion window.

The D-SACK is an extension to the SACK option [101] enabling the receiver to accurately report the reception of duplicate data. The first block of the SACK option field can be used to report the

sequence numbers of the duplicate data that triggered the acknowledgment. This D-SACK allows the TCP sender to infer the order of packets received at the receiver. This extension is compatible with current implementations of the SACK option in TCP. Even though one of the TCP end nodes does not use this D-SACK extension and the other TCP end node does, this use of D-SACK by one of the end nodes will not cause problems. In other words, the TCP sender that does not understand this extension to SACK will simply discard any D-SACK blocks and process the other SACK blocks in the SACK option field as it normally would.

2.6.10 Header compression

The lossy link characteristics may harm the end-to-end TCP throughput in the sense that higher link error rate would less ensure the probability of successful transmission of packets. Thus the header compression may improve the TCP throughput over the error prone link to some extent, but should be carefully considered the efficiency of the use—if a reliable link-level retransmission strategy exists, the end nodes may not suffer from the wireless corruptions because the link corruptions could be locally managed to be recovered, and in addition, in use of header compression TCP end nodes may suffer from time spent due to the considerable effort of decompression required. Nevertheless, due to multi-hopping relay nature, in case of long hops traversing, the header compression may be challenging to an extent for reducing RTT by lessening transmission delay; furthermore, reducing the overhead is especially important for connection with a small MSS. However, the header compression cannot be applied to segments having the timestamp or SACK TCP option and thus is not applicable to our study because we use both options.

Chapter 3
The Rationale of the New Scheme

3.1 Introduction

A reliable routing protocol for ad hoc networks is selected either on the basis of its characteristics to find a shorter route (by means of table driven or by on-demand) or to maintain a long-lived route (discovered by means of packet signal strength [51, 69, 70]). Sometimes multiple routes or a different route for the return path might be preferred. This fact means that transport protocol should act resiliently with different type of routing schemes employed at aiming towards a promising behavior (i.e. specific levels of node mobility, traffic scenario, network size, etc).

On the other hand the MAC protocol is chosen to share the network resources with fairness over competing nodes (while handling the so-called channel contention-induced congestion) or to improve the link efficiency in aggregate throughput. Beside these characteristics, the MAC protocol has a local retransmission mechanism performing a certain number of attempts when the link is highly contended or error-prone.

Conventional TCP is not able to fully utilize the available bandwidth provided by the lower layer protocols because the normal TCP end hosts do not utilize inter-layer and route-specific information provided by MAC and routing protocols; D. Sun and H. Man [138] addressed the drawbacks of several TCP variants (e.g., Tahoe, Reno, NewReno, and SACK) in the mobile ad hoc networking framework.

In this Chapter, therefore, we introduce elaborate TCP receiver and sender which can perceive the present condition of the network (at the beginning of the session establishment or at another time) by adjusting itself rather than consuming the network bandwidth.

3.2 Prerequisites for the proposed scheme

The prerequisites of proposed scheme are clock synchronization, inter-layer vertical information delievery and path MTU discovery, which are explained in the following subsections.

3.2.1 Clock Synchronization

In ad hoc networks the receiver is required to keep track of the forward link delay (FLD) that coarsely indicates the capacity. It informs the sender about a maximally allowable bound on the bottleneck capacity. The clock synchronization between communicating end hosts is thus necessary for computing the meaningful FLD, where the arriving packets can provide an accurate estimate of FLD by using the timestamp option provided by the synchronized clock.

Due to the small size of ad hoc networks, the nodes might not face hardship in synchronizing configurations. One possible passive procedure for participating mobile nodes (after a three way handshake), is to take the useful information from the link condition, such as forwarding capacity allowed by power levels, passive coarse clock-synchronization mechanism and passive RTT measurement.

A necessary mechanism for proper operation of the proposed scheme is a coarse clock synchronizing strategy (Appendix A.1). . It is used for the following purposes in ad hoc networks:

- To compute the maximally allowable bound capacity determined by the FLD (measured by the arrival of a synchronized time-stamped packet in support of the clock synchronization mechanism).

- To find the worst link condition as determined by FLD and its deviations (measured at the receiver), which are used for deciding when to propagate the feedback to freeze the sender.

3.2.2 Inter-layer vertical information delivery

In order to attain the newly introduced flow control mechanism, the number of hops which a packet traverses must be informed to the transport layer each time it arrives. The hop length of a route may be updated (or maintained if the hop length is the same as before) every time the route has been

partially or entirely rerouted; however, the inspection of TTL value in each of the received data packets can not signify all switches of path link.

The way to inform the number of hops used can be a method which involves the checking of the routing overhead of the routing control packet or, by incrementing a counter (or decrementing TTL field, which is usually set to 32 by default in ns2) in the IP header of each packet to let intermediate routers increase it at each hop. In this thesis, decrementing TTL is used so that the receiver can realize the number of hops between end hosts.

3.2.3 Path MTU discovery

Delay-Bandwidth product [119] defines the channel capacity to determine how many bits fit in an imaginary pipe of data. The length of pipe corresponds to the latency, diameter to the bandwidth, and delay-bandwidth product to its volume or number of bits it can hold.

Each packet transmitted from a TCP sender imposes constraints on the bandwidth required by the overheads of lower layers. Thus, at the receiver side the number of packets received implies a considerable amount of bandwidth consumed by the overheads of lower layers.

The proposed receiver is elaborated to compute the next appropriate sending rate by regarding the lower layers' overheads. These overheads specifically depend on medium contention, node mobility, etc. Therefore, rather than the window advertisement based on byte-stream, packet-stream based advertising will beneficially delimit the available times for packet transmissions in multiple units of a specific-sized packet (i.e., control the frequency of medium access, in the sense that a consistent amount of MAC overheads is required unless medium contention is given). Hence, the sender can strictly control the transmission window equivalent to the amount of bandwidth required for both a single data transmission and the lower layer's overheads.

In order to achieve the packet stream-like byte stream advertisement between end hosts, the sender and the receiver should comply with the following:

- As in RFC 1191 [103] allowing the largest possible packet size between two end hosts, a sender should implement the PMTU discovery by setting the "Don't Fragment" bit in packets (i.e., IP datagrams) and by reducing the packet size according to ICMP

"Destination Unreachable" control messages from intermediate routers when necessary. The sender then transmits in multiple units, with the DF bit set, of a specific-sized segment [1](i.e., the MSS option with an upper bound of (PMTU – 40 bytes of a default TCP/IP headers)) determined and occasionally reduced by a subsequent PMTU discovery.

- Complying with that, once the receiver advertises, the amount of advertised bytes should be equal to the truncated amount being equivalent with multiple units of the MSS observed. Once a receiver advertises a zero window, it should delay advertising a nonzero window[2] until it has at least one MSS equivalent bytes (i.e. one packet) available.

The following subsection will address the modifications at the receiver end, which will occasionally limit the sender's effective window by comparison with the receiver's advertised window. The receiver's advertised window will supersede the sender's spurious congestion window progression, as *an ad hoc-adaptable medium contention avoidance mechanism*.

3.3 Modifications in Receiver-end

After studying characteristics of ad hoc networks, we conclude that both sender and receiver ends are required to be modified for optimized performance. The major modification proposed is required to be made on receiver-end[3] because of the nomadic nature of mobile users. As these mobile users usually operate in receiving mode (rather than sending) while interconnected to the wireline Internet. In the next subsection, we give justification for use of advertised window in the newly proposed mechanism.

3.3.1 Justification of use of the Receiver Advertised Window

In an ad hoc network, a nomadic mobile node may communicate with another network, which may not know the peculiarity of the ad hoc network set up.

[1] MSS only applies to the data payload.
[2] If a receiver advertises in byte-stream basis as soon as it becomes available, it may cause behavior known as the silly window syndrome. Silly window behavior [43] is characterized as a situation where the receiver's window oscillates between zero and a small positive value. Such behavior leads to inefficient network utilization because each segment has to contain little data compared to the necessary overheads for lower layers.
[3] Likewise, TCP REAL [149] is a receiver-equipped congestion control algorithm that introduces a fully receiver-oriented congestion control algorithm with congestion avoidance and advanced error recovery tactics and aims for heterogeneous environments with wireline or wireless networks and delay-sensitive or –tolerant applications.

As the nomadic node in the ad hoc network behaves as a receiver, it is reasonable requirement that receiver informs sender of an optimum sending rate in relation to the current network dynamics. If the nomadic node is behaving as a sender, then it shoud know an optimum rate for sending data and a proper rate control mechanism in the given network conditions.

The reason for these requirements is that the ad hoc links change dynamically with time due to unpredictable node mobility and competition among mobile nodes for sharing a single common medium on random access basis. Thus, in the case of a connection between nodes in an ad hoc network and nodes in another wireline network, the sending rate[1] should reflect the conditions of ad hoc network which should be dominated at bandwidth-constraints of the associated links.

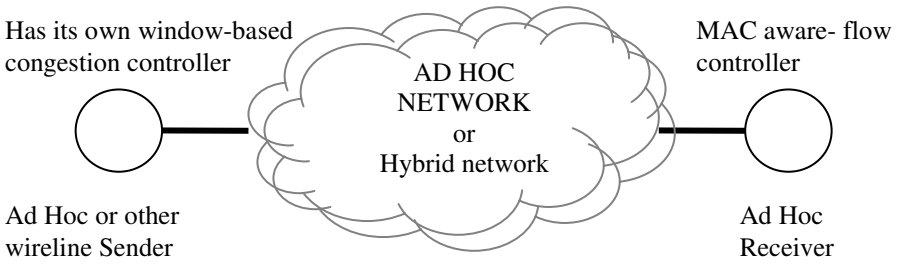


Figure 3-1. Controller equipped at either end

In order to reflect the network link constraints, as shown in Figure 3-1, the elaborate *flow controller* of a MAC-aware congestion controller is equipped at the receiver-end in the sense of the frequency of the nomadic node behaving as the receiver; therefore, the receiver now takes into account two things, the path link condition perceived and ordinarily the receiver's buffer space to advertise its *receive window*. The use of the receiver's advertised window is justified because existing TCPs, such as TCP Reno or TCP Vegas[2], are restricted by the receiver's advertised window, in principle that it basically prevents the sender from overrunning the receiver's buffer.

[1] Resulting from the dynamic nature of an ad hoc network, instantaneous rate (throughput) available at a router, hopefully estimated at the receiver-end, will be a good metric to determine extremely time-variant link condition and will be in turn translated accordingly to the sender's window size because TCP is not a rate-based, but window-based transport protocol as detailed in the chapter 3.4.7.3.

[2] Well-known proactive based congestion control algorithm, which is responsive against additional buffer queued in the network path, determined by RTT variation observed at the sender-end. If the buffer queuing up goes over the upper threshold, the sender detects that the network has currently reached incipient congestion stage, and reduces the sender's window in a certain manner (addictive decreasing phase). If below the lower threshold, the sender increments addictively. In between, it maintains. Here, the use of two thresholds should be dynamically adjustable for use in the ad hoc network.

In particular, because a conventional reactive-based congestion control mechanism increments the congestion window until it encounters a packet loss and thus from time to time overoccupies the network medium (resulting in increasing *MAC related delay*), the elaborate receiver limiting the sender's transmission window (namely, *congestion window delimiter*) could play a significant role in controlling, so-called, medium access frequency. Consequently, the use of the receiver's delimiter can avoid unnecessary slow start phases to an extent, which produces a significant performance gain [139] over the lossy contending environment. The following subchapters will detail and evalaute the *congestion window delimiter*.

3.3.1.1 Receiver based perception of the network capacity

In most of existing TCP implementations, the sender adjusts sending rate and invokes congestion control in response to ACKs (and/or timing information it obtains by measuring round trip times). In the meantime, the receiver advertises its buffer level, which is, in general, higher than the sender's congestion window. During the whole connection time, the sender is mostly *congestion window-limited*. However, in the sense that the receiver could have a more precise knowledge of the traffic, especially of the forward link, it seems more reasonable that the sender has to be *receive window-limited*.

MaxWindow = MIN (CongestionWindow, AdvertisedWindow)
EffectiveWindow = MaxWindow – (LastByteSent-LastByteAcked

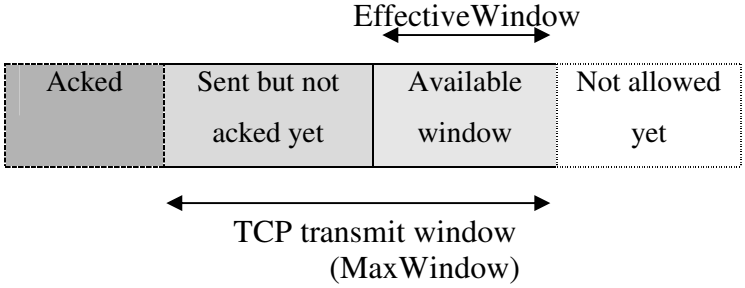


Figure 3-2. TCP transmission window

As a consequence, both windows restrict the effective window as shown in Figure 3-2 [119]. When the advertised window is less than the congestion window, it means that the sender is now under the control of the receiver. Due to the intentional limitation of the receive window, occasionally it yields a negative effective window. In that case, the sender will not transmit any further packets (frozen) until the effective window is greater than zero, where it is called a *provisional frozen*.

In normal TCPs, when a packet loss occurs, the sender reduces the congestion window to either one or half, at which the effective window becomes negative in general and further timeouts can shrink *ssthresh* and discard all in-flights. Likewise, in the case of the negative *effective window*, the sender should cancel the retransmission timer to the effect that it could prevent spurious RTO-induced packet loss from the postponement of ACK arrival because the negative effective window implies a heavy contending situation that causes late ACK delivery.

Within a certain time, a subsequent ACK with a window update will invoke further transmission (because it convinces that the route is still being connected). Then, the sender sets the timer again if further in-flight(s) still remain. Furthermore, when the modified sender is applied, this functionality will give another chance to reassess the current inadequate set of *ssthresh* and to make it more appropriate because the sender is modified to take a new *adwin* received and used as a new *ssthresh* each time it has the negative effective window. Chapter 3.5 further details this functionality.

In brief, the receiver in an ad hoc network is therefore designed to estimate the best rate for the best throughput, taking into consideration versatile metrics, which is capable of reflecting the currently dominating link condition. Then, periodically, the receiver informs the sender of the available window size, by designating in the receiver's advertised window field according to the best rate computed.

3.3.1.2 Limit the maximum congestion window

The *congestion window delimiter* to inform of the optimum window reflecting changing network condition probably restricts the sender's effective window and puts the sender under the control of the receiver. In the case where the receiver delimits the congestion window and fortunately mitigates packet drops to a considerable extent, the sender progresses its congestion window (*cwnd*) incessantly each time an ACK is received. That is, under the receiver's limitation, the sender's congestion window usually gets overblown.

As seen in Figure 3-3, for instance, the receiver advertises (unusually) one packet as the advertised window (*adwin*) over a 5-hop stationary path with extremely heavy traffic loads from 20 to 30 sec. It occasionally blocks even one packet transmission, so timeouts. In general, owing to being under the control of the receiver advertising *adwin* to 1, the sender is not likely to experience packet loss

for most of the connection time, so it results in the constant evolution of the congestion window, here inflated up to 28 segments.

From the figure, TCP sender was initially set to 20 for slow start threshold (*ssthresh*). The *cwnd* increased until timeout occurred at 21 sec, at which the *ssthresh* was reduced to 2 as the minimum *ssthresh*. Afterwards, it has suffered from the exponential backoffs postponed till 35 sec.

At 35.2 sec, the sender was put into the congestion avoidance phase. In the meantime the effective window was still set to 1 by *adwin*, due to which *cwnd* inflated again. On an occasion, in terms of such discretionary *cwnd* inflation, tactically a malicious user (greedy receivers) may use to drive a high *ssthresh* set by means of simply setting the *adwin* back to an overblown value. Therefore, the sender should set a maximum congestion window (i.e., "maxcwnd_" in ns2) size to a certain value just in case.

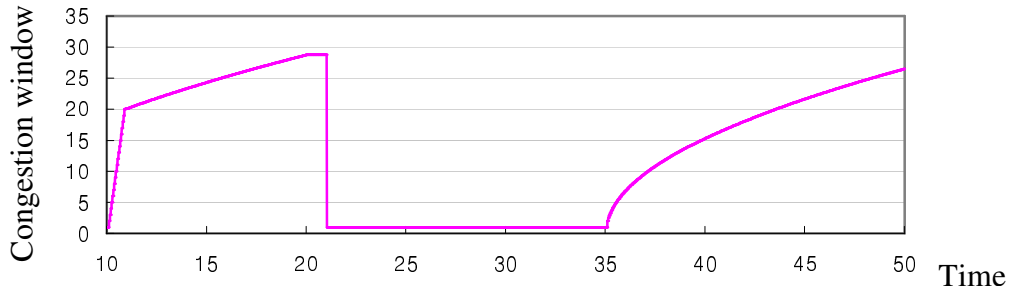


Figure 3-3. Congestion window with advertised window set to 1. It shows the receiver's window-limited situation causing the sender's congestion window inflation.

3.3.1.3 Recognition of the wired sender and receiver

Continuous increments of the sender's congestion window when it is receive-window limited might passively discriminate ad hoc receivers from wireline ones because in general the wireline receiver advertises relatively higher window equivalent with receiver's buffer. Thus, the ad hoc sender can behave differently according to the type of receiver (see Appendix A.7). On the other hand, in terms of recognition of the wireline sender, items of concern are addressed in Appendix A4. In the case of an ad hoc receiver and its wireline sender, communicating with each other and, meanwhile, having a packet loss in part of the wireline network, the congestion window of the sender is shrunk to a small one and the receiver no longer restricts the sender's congestion window until it progresses and exceeds the receiver's advertised window again.

3.3.2 The uses of the receive advertised window

The following examples, in view of limiting sender's transmission rate, introduce mechanisms that inform of the network resource availability in respect to bandwidth-constraint path condition either determined at network, or estimated at end-receiver.

J. Semke et. al. [131] proposed the automatic TCP buffer tuning mechanism. It dynamically adjusts the socket buffer size of either sender- or receiver-end in response to the network path condition determined respectively by the congestion window or the receiver buffer usage at times of packet absence in sequential order. However, this scheme basically aims for wireline networks.

In principle, in terms of the receiver's buffer adjustment, it detects whether the current data rate is reduced by a small receive window or a slow bottleneck link. For example, in case of packet loss, the receiver's socket buffer is queued up waiting for reassembling the data stream in sequential order, and in the mean time the receiver throttles the sender by reduction of the receiver socket buffer whose available space is equivalent to the receiver's advertised window; otherwise, the buffer size is increased. The buffer size will still calibrate itself when it detects a packet loss as long as the buffer is much larger than the space required during the recovery of the packet loss.

Cited in [81], L. Kalampoukas proposed the Explicit Window Adaptation (EWA) mechanism to control the sender's transmission window by means of an explicit window feedback designating the state of the buffer of bottleneck routers along the connection path. EWA modifies the receiver's advertised window size, which is then returned to the sender and controls accordingly with the state of buffer level informed. The feedback conveys the appropriate window size based on the amount of buffer space. EWA makes the advertised window play a role in avoiding buffer overflow and maintaining at a certain level below the buffer capacity.

In Freeze TCP [68], a mobile host predicts wireless channel disconnection using link-layer information available from MAC protocol employed and then returns zero window size to a sender in order to freeze the sender.

Similarly, the study in [16] proposed a TCP rate control scheme that adjusts TCP receiver's window. It estimates the router's advertised window size associated with fluctuation of the router buffer level and then reduces the receiver's advertised window size if it exceeds the computed feedback value.

A-TCP in [132] introduces a TCP rate control scheme functioning at base station where the advertised window field of ACK can be modified to inform its sender (a mobile host) of the channel capacity of the base station as well as channel disconnections. The mobile host in turn decides a proper window size according to the channel status, which is not bigger than buffer size. The A-TCP estimates the channel capacity by means of monitoring the local retransmission buffer in base station because packet losses due to BER, channel disconnection or hand-off must reduce channel bandwidth and increase the number of packets in the retransmission buffer.

Indeed, the control of the advertised window requires intermediate routers involved because the end-to-end based perception of the network condition hardly select an optimal value in order to avoid contention or buffer overflow along the path link, and thus the involvement of routers is strongly necessary to determine accurate network condition. For example, EWA utilizes intermediate routers along the connection path, in which accordingly the receiver provides an explicit window feedback of the state of the buffer level, and so all the routers have to be equipped with the same window feedback capability; however, it becomes a drawback in terms of the entire deployment of the routers' functionality.

Likewise, the Bandwidth Aware TCP (BA-TCP) and Rate Adaptive TCP (RA-TCP) proposed in [66] and [84] respectively also address the network-originated information such that BA-TCP requires adding extra information in packets passing through routers at every instant to deliver the propagation delay and the available bandwidth information to TCP receivers, and RA-TCP facilitates an explicit feedback mechanism to convey the fair session rates from the network directly to individual TCP senders.

These kinds of mechanisms using the network-originated feedbacks to help with deciding the next available bandwidth, beneficially reflect more accurate rate controls, compared to the network-independent end-to-end based rate controls, but they need dedicating bandwidth-consuming routers to function for end hosts and be responsible for fairness and effectiveness among concurrent connections.

In ad hoc networking environment where all nodes are freely and temporarily configured on a specific purpose, any node does not want to be elected spontaneously to perform as such an intelligent, centralized base station to serve for others even though some papers introduce network-

oriented control mechanisms dependent on elaborated routers in the network (i.e. TCP-Bus [86], TCP-EXACT [42], ATP [139]).

Most enhancements addressed above in terms of handling wireless links requires the network-wide elaborations like the involvement of path routers and bottleneck base station. In the sense that the scheme motivated by this thesis will not depend solely upon the network-originated information, the following Chapter introduces the charateristics of the wireless medium and verifies the likelihood that the network path link condition can be perceived by the end receiver.

3.3.3 Tailored to the MAC constraints

To stimulate the feasibility of the receiver-oriented flow control in the context of ad hoc networks, the study pointed out in [63, 153] has interestingly shown that, there exists an optimal value[1] for TCP congestion window size that maximizes TCP performance. That was verified over several typical ad hoc networks (i.e., chain, cross, and grid) in which 802.11 MAC protocol of our concern is the main restriction that determines available bandwidth. Zhenghua Fu et. al. [63] argued the likelihood of, and justified the need of, a MAC-aware congestion control mechanism according to the MAC related delay and routing errors[2] because in 802.11 MAC framework (over a series of simulations with typical multi-hop wireless paradigms), all losses were caused by medium contention; consequently, it states the inefficiency of the 802.11 MAC protocol in this case. However, it disappointed in that it would be hard to present optimal window values in randomly changing topologies as expected in real ad hoc situations.

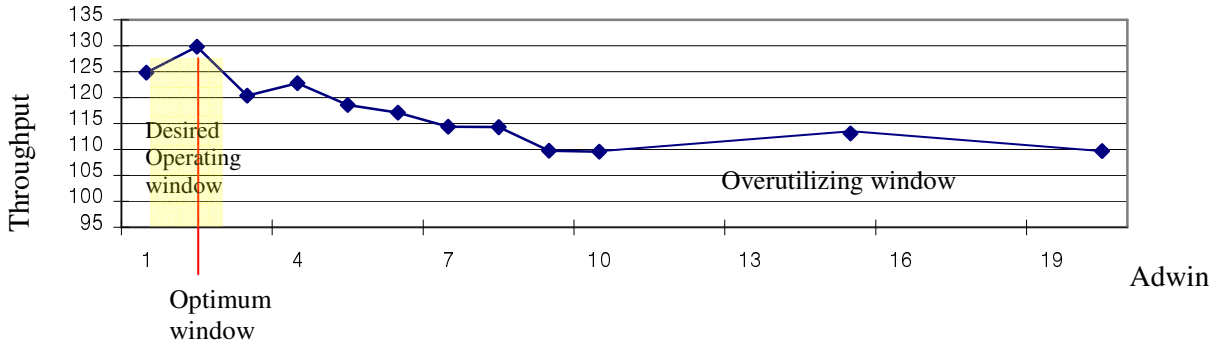
Table 3-1 below shows TCP throughput gain beneficially obtained by changing *adwin* when experimented over 5 hop stationary sting topology. It was best at a*dwin* of 2, by 18.3 % over at of 20 at which the sender is solely *congestion window limited* for the whole connection time.

[1] Normal TCP (new Reno) does not operate around this optimal point to end up with sub-optimal behavior resulting in degraded throughput and increased packet loss because TCP is originally conceived for use in wireline networks where all loses are almost implying buffer overflow in a bottleneck router and its flow control is supposed to be responsive with buffer overflow to invoke congestion control algorithm. Thus, the normal TCP does not work properly over wireless networks where spatial channel reuse is a more important concern than the resolution of buffer overflow.

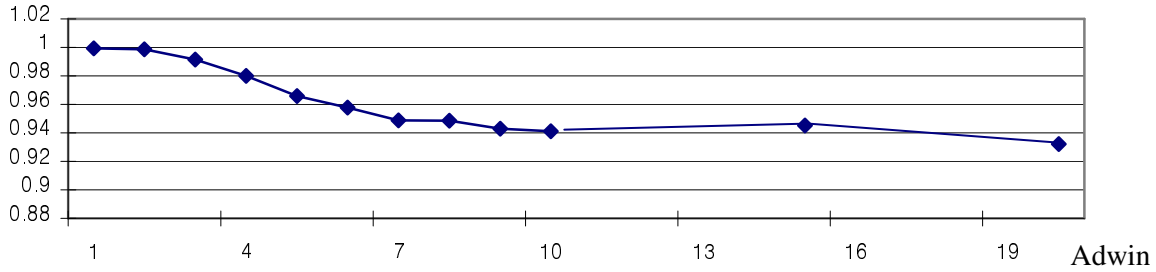
[2] Most routing protocols facilitate the use of MAC layer detection to determine the viability of route established. Thus, substantial MAC delay may imply an imminent route breakage.

Adwin	1	2	3	4	5	6	7	8	9	10	15	20
MaxSeqno.	1597	1661.2	1541.7	1571.7	1518.3	1499	1464.7	1463	1406	1402	1448	1404
Fast Ret.	0	0	0	20.7	34.5	40.7	42.7	44.7	39.7	38.3	38.3	37.7
Timeouts	0	0.8	10.3	3.7	3.8	2	3	4.7	4.7	6	4	7
Throughput (Kbps)	124.8	129.8	120.4	122.8	118.6	117.1	114.4	114.3	109.8	109.6	113.1	109.7

Table 3-1. Throughput improvements made by limiting advertised windows. 5 hop stationary string topology with TCP Sack, active for 100 seconds and its packet size 1000 bytes.



(a) Throughput according to a limiting *adwin*.



(b) Goodput according to a limiting *adwin*.

Figure 3-4. Throughput and goodput according to different advertised windows for 5 hop stationary string topology with TCP Sack (From Table 3-1).

As seen at *adwin* 3, its throughput was degraded by having many timeouts, rather than fast retransmits, due to insufficient in-flight packets when a packet loss occurred, so was slightly less than that at *adwin* 4 where most of packet losses are recovered by fast retransmits. However, as evident in Figure 3-4 (a) and (b) for the other higher *adwins*, throughput and goodput were degraded because heavy medium contention was induced by a single TCP connection itself (i.e., sender's transmission window overwhelming the available network resources), where the forward path is identical to the reverse path (i.e., fundamental *self collision* problems [152, 156]), and in spite of none of other competing traffic sources, TCP sender suffered from a number of fast retransmits and timeouts. That is, as desired, dynamic adjustment of *adwin* according to the degree of present

medium contention will improve the throughput; besides, if other traffic sources were applied, throughput gain can also be achieved by a simple limiting *adwin* in comparison with the absence of limiting the *adwin*.

As a consequence, the perception of existence of the optimum window size encourages the investigation of the optimum window adjustable according to the network path link condition, in order to achieve the best TCP throughput and *goodput*—with an increasing number of back offs or local retransmissions, due to high contention or other reasons such as link corruptions or routing failures, the receiver should limit the sender's transmission rate in order to prevent building up queue and to mitigate the level of contention.

3.3.3.1 Reference metrics identifying ad hoc path links

As the dynamic peculiarity[1] of *ad hoc* situations, the network resources are underutilized by erratic, undesirable behavior due to sub-optimal interactions between layered protocols. Many researchers have addressed TCP throughput in terms of interactions with a MAC or a routing protocol but hardly formulated TCP throughput. For example, according to the number of hops apart or concurrent transmitting nodes, each node has the specific ranges of transmission and interference that could vary, of course, due to the level of battery power, and sometimes nodes are unevenly interfering with each other, and their position or movement also affects differently to another. Delays in total contributing to such diverse uncertainties can be hardly characterized so as to clarify the degrading factors rigidly and then to make a reasonable rate change.

Therefore, to build the receiver-end flow controller, we should first keep in mind *reference metrics* to identify certain path links: it could be, for example, the number of hops of the path, a moderate level of battery power required in each involver, least contention level from other third party interfering nodes determined by a specific per-node interfering range, and density and mobility of involving mobile nodes if perceivable—Later, some of the reference metrics will be addressed for use, such as a minimum FLD, least contention level and least *buffer occupancy* (each metric is available for a certain hop length), and *attenuation factor* that is characterized by a specific lower layer implementation, such as 802.11 MAC protocol. Under the path condition of the *reference*

[1] In brief, ad hoc link capacity is varied by node mobility causing fading, shadowing, link breakages, or long-term network partitioning (inducing higher routing packets flooding), and interfering nodes (causing serious MAC contention delay or sometimes blocking), resultant queuing delay (overflow), and relatively high BER.

metrics, the receiver can then monitor and reckon the level of performance degrading factors when exposed to additionally induced *MAC-related delays* (Chapter 3.4.2.1)

3.3.3.2 Router-elaborated determination of the hop link capacity

Instead of the end-to-end based analysis to identify the MAC-related delays, other approaches (e.g., [42, 63, 139] for TCP, or INSIGNIA [87] as an QoS in-band signaling) for a particular use in ad hoc network are based on router perspective to determine each hop link capacity at an instant of packet forwarding. It is, of course, more accurate than the end-to-end approach. However, it needs more intelligent routers to compute the available link capacity, regarding hop-based medium contention, and to deliver the computed information to either of end hosts.

As a router based approach in relation to a medium contention resolution scheme, Zhenghua Fu et. al. [63] proposed two schemes; the distributed Link RED (LRED) algorithm and Adaptive Pacing algorithm, in the sense that the elaborated router can give an incipient indication of medium contention induced congestion rather than buffer level related congestion.

In the LRED algorithm, the packet dropping probability at each router does not vary according to the queue level of the router but to the degree of medium contention. By awareness of that the link drop characterized by 802.11 framework exhibits a behavior similar to RED over the Internet, packet dropping probability is computed as a function of an average number of the retires of recent packet transmissions (i.e., the occurrence of hidden terminal drops means the presence of network overload).

Hence, LRED algorithm play a role in reducing medium contention gracefully in terms of the packet dropping probability, which accordingly provides ECN-bit designated TCP flow to simply inform the TCP sender of network overload.

The Adaptive pacing algorithm, as the second technique proposed, aims to prevent the traffic from overloading the network, working in parallel to the LRED algorithm. When packets traversing through a downstream link contend for the channel, a packet might be dropped by a concurrent transmission at one hop-away node (as the well-known exposed terminal problem). So, the extra back off interval will be added into a random back off period so that it mitigates the contention drops from the exposed receivers (i.e., the random back off, plus one packet transmission time that

extends the range of the link-layer coordination from one to two hops along the packet forwarding path). As a result, the addition reduces the likelihood of contention drops caused by the exposed receivers.

In brief, these MAC layer-based two approaches functioning at each router, therefore, try to tune the packet dropping probability as a function of the number of local retransmissions (i.e., a number of hidden terminal packet drops due to high contention or of higher BER), and to mitigate the likelihood of occurrence of the exposed terminal induced drops by adding an extra back off period (i.e., effective spatial channel reuse is attained). In simulation with Newreno and Vegas TCPs, they can enhance the TCP performance considerably by 5% to 30%. Beneficially, LRED router based approach can allow unmodified TCP senders to respond with.

In the sense that the medium contention interferes widely over the adjacent links and its availability fluctuates due to random medium access interdependency with the backoff and the NAV among current competitors, the following Chapter details the rationale of the receiver-end based bandwidth estimation.

3.4 Envisioning Network Capacity at the receiver

The need of the flow control mechanism to respond accordingly with MAC related delay as well as frequent routing errors (disconnection) has been justified in the previous Chapters. Now, the following gives an insight in estimating the network bottleneck capacity and the boundary extremes determined by reference metrics of the path identity, all of which are obtained at the receiver-end.

3.4.1 Inadequacy of current end-to-end bandwidth estimators

In terms of the flow control originated by the receiver, it has to estimate the available network capacity that could imply the level of medium contention and predict impending route failure if possible. The short-term throughput of incoming packets is simply measured by the number of received packets divided by a certain sampling time interval, which could approximate the network capacity available at every time instant. However, ad hoc links are of transient nature and sometimes have inconsistent intermission times (i.e., packet jitters) between packet arrivals due to instant path switches or other anomalies, such as medium contention suffering. Thus, the following

end host based bandwidth estimation techniques below may not be practicable for direct use in the ad hoc networks.

Clark and Fang [44] present a measurement based algorithm, namely Time Sliding Window (TSW) algorithm, which calculates the instantaneous sending rate. The computation does not rely solely upon packet size divided by inter-packet arrival time but accounts for a certain interval worth of past history in the manner of sliding a time-window (namely, time-window worth of past history). Then, it gives a more reasonable rate computation such that, the sending rate is computed by received bytes divided by the time interval and so smoothes out the burstiness of packets. Hannan [73] used this TSW algorithm to estimate the available rate for ad hoc links and contingently to alleviate the impact of the highly fluctuating packet jitter problem resulting from the interdependency of medium access.

However, there is a point to be concerned about. TSW rate computation algorithm can determine the network bandwidth availability, only supposing that the sender continuously transmits data (bulk data transfer). But, in practice, due to the fact that data originations are basically restricted by the medium availability in terms of common shared medium and, more considerably, susceptible route (i.e., long intermissions between packet arrivals), it hardly validates the rate computed to fully determine the available link bandwidth.

And, as another estimator, namely, SLoPS as in [78] has to sample arriving packets to compute the available bandwidth in a way of moving average manner taking into account the variation of the inter-arrival times of packets (i.e., smoothing out the variations). Thus, the estimated bandwidth may be validated historically by previously sampled packets but not be instantaneously determined by a single recent packet received. Although the use of the SLoPS may be applicable in stable path links routed, such a historical sampling way will not be adequate for a direct use in instant time-variant ad hoc links dynamically rerouted.

3.4.1.1 Our Propositions

Therefore, we propose a new rate computation algorithm, which is most adaptable to dynamic ad hoc links (for the purpose of limiting the sender's effective window) and is robust to the impact of the jitter effect in packet arrival times (the burstiness or the starvation of ACK arrival).

Simply from the use of forward link delay and its variation between packets, two control aspects will be contemplated:

- Explicit window advertisement by the mean level of the instantaneously estimated bottleneck throughput determined by the forward link delay and its variation in respect to path identity.

- Explicit freezing mechanism by the mean deviation of the forward link delay, which determines the level of medium contention, in the sense that the higher the medium contention, the higher fluctuation.

At first, the following explains the instantaneous rate computation way to give a window advertisement accordingly with path link delay. Then, a subsequent Chapter will give an explicit freezing signal of ZWA each time heavy medium contention is perceived at the receiver.

3.4.2 The Effective Bandwidth

In the receiver-side, it endeavors to estimate the available network capacity and gives periodical feedbacks to inform a correspondence of leftover network resources. A reliable flow rate derived from a right dimensioning of the network capacity at any instant aims in order not to overwhelm the network capacity.

It seems to be a limitation to dimension the leftover network capacity over multi-hop links at the receiver side and to mimic the network capacity available at a bottleneck router by means of metrics obtainable at the receiver. If the bottleneck bandwidth is determined and, as accurate as possible, the receiver can ensure to some extent that, it can control the sender accordingly with the receive window advertisement.

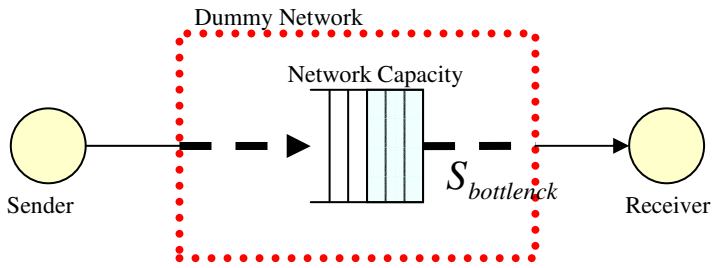


Figure 3-5. Network Capacity characterization –Inside the network, nominally, designs a queue representing the network capacity characterized initially by the number of participating routers and then by induced medium contention or node mobility.

As shown in Figure 3-5, the queue of a composite router superseding participating router(s) along the path, does not embody an actual queuing level of packet heaps but implies the instantaneous bottleneck throughput ($S_{bottlenck}$) signifying network resource availability at a bottleneck router along the path, for which in majority its capacity is determined by widely influencing co-channel interference varied. An analytical model with mathematical evaluation in microscopic view of the composite queue is available in [22]. It addresses the bandwidth estimation technique through observations at end hosts but is only in relation to the measurement of packet service time of the bottleneck router of the connection, and therefore is not directly applicable onto ad hoc links—because for wireless links a MAC protocol is working that is more problematic to determine the available bandwidth.

Therefore, particularly on ad hoc links, the bottleneck throughput[1] implies the outgoing rate degraded by the degree of medium contention delay (characterized by the random access based 802.11 MAC framework), transmission and propagation delays, route breakages (i.e., a number of medium access failures), and wireless link corruptions as well as actual queuing delay. Transmission and propagation delay is invariant when a fixed packet size.

In a network where there are a centralized base station that can carry out spatial channel reuse over contending mobile users in a way of contention-free time slot allocation[2] discovering the optimum

[1] As conceptual distinction between terms, *throughput* and *bandwidth*, the term *throughput* defines the number of bits per second that can be transmitted over the link in practice, measuring performance of a system with a specific implementation, while the term *bandwidth* means the raw link capacity available on the link [119].

[2] It should require an access point (AP) functioning as a hub, centralized scheduler to attain contention-free and time-bounded channel allocation. Practically, so-called Point Coordination Function (PCF) or a newly introduced hybrid approach (HCF) can not be however employed over an ad hoc network because of the

time schedule (e.g., time bounded MAC scheme referred in [47]), so-called effective bandwidth ($B_{effective}$) can be maximized so that the leftover bandwidth is fully available when to send out data. However, the lack of the supervising base station does not allow competing nodes to share optimisingly with each other—it means the effective bandwidth allowed for each transmitting node relies upon the performance of MAC protocol employed.

Several studies have endeavored to explain the effective bandwidth over specific ad hoc cases in an analytical manner such that, Zhenghua Fu et.al. [63] determined the effective bandwidth that carries out optimally a spatial channel reuse in specific topologies. And, Jinyang Li et. al. [88] further examined the capacity of ad hoc networks in diversity of topology and traffic load and accordingly evaluated packet forwarding ability, scaling behavior of per node capacity. The study presented the effective channel capacities in several typical topologies and conclusively argued that the feasibility of scaling the network size depends on traffic patterns applied.

And, Xiao Hannan [73] introduced namely a Flexible QoS Model for MANETs (FQMM) as the first proposed QoS model for ad hoc networks. It presents both parameter- and measurement-based calculations to characterize the effective link capacity over dynamic moving nodes, by means of subtracting the aggregate bandwidth being used by adjacent neighbors (of one or two hop away transmissions) from the raw link bandwidth (i.e., line rate)—with the assumption of no congestion occurred due to buffer overflow (i.e, assuming enough buffer spaces) at any nodes along the path, and of no random wireless loss of packets made once the medium is occupied.

As a result, the effective bandwidth determined is likely to be transiently validated due to the intervention of interfering nodes unpredictably. Thus, the measurement of the effective bandwidth requires non-trivial effort to validate at every instant.

The following section defines the type of delays each packet experiences when traversed through the path link, and then the next Chapter explains the hop-based link throughput characterizing the maximally allowable bound network capacity.

---

absence of such an integrated station to configure the time-bounded services. Thus, in reality, the fundamental DCF mechanism is the only way to access the medium thus yields undesirable MAC contention delays.

3.4.2.1 Classification of delay type

Upon TCP data transmissions, a specific data packet originated from a sender will experience substantial delays, such as queuing, medium contention, transmission, propagation and processing delays.

So that the amount of delay implying the instant network delays could be computed in advance, the followings define delay types one packet (i.e., IP datagram) experiences when it passes (incoming and then outgoing) through each router.

Queuing delay corresponds to the total amount of delays relating to dequeuing of other packets already been queued in advance. Time taken for either queuing or dequeuing packet (time taken to drain out a packet) at a router is dominated mostly by medium contention delay. So, the increasing queuing delay could be caused by intensive medium contention present—the medium contention occurs within the RTS-CTS handshakes, and its resultant delay is proportional to the number of neighboring competitors. So, comprehensively, the medium contention related delay will dimension queuing delay because the medium contention related delay [1]can be roughly equivalent with the queuing delay in the sense that a smaller delay of medium access may result in buffering less packets and servicing quickly, and vice versa (As a matter of fact, the precise queuing delay is very hard to be obtained from mobile users that use diverse mobile computing devises). Thus, total amount of queuing delay could render instant network loads induced by other traffic, such as other contending traffic sources, control frame exchanges, routing control packets flooding, or retransmissions due to channel errors.

Transmission delay for an IP datagram dequeued is characterized by the line rate (raw link capacity, denoted as B_{raw}) and the packet size, as shown in Figure 3-8, assuming negligible *processing delays* related to *packetizing* between layers.

Propagation delay is applied to all packets as equal, independent of the packet size, and involved as the *comprehensive MAC related delay* because the next transmission can be available after the reception of the additional MAC ACK frame. And, in computing rate at the receiver by means of

[1] Comprehensively, it involves many contributing factors, such as back offs that mitigate the likelihood of collision or capture effect, and NAVs, and time taken for RTS/CTS/DATA/ACK exchanges including inter frame spaces, and occasionally additional time taken for local retransmissions triggered by wireless link corruptions.

FLDV as proposed later, the inter-packet delay difference of the FLDV is independent of the identical propagation delay.

From the types of delays reviewed, all the delays required to drain out a packet from the head of queue is called the *comprehensive MAC related delay*, as the term used before. It includes queuing, medium contention, and propagation delays because each of these delays cannot be explicitly determined by the TCP receiver except the transmission delay. Beneficially, the end-to-end bandwidth estimation accounting for this *comprehensive MAC related delay* mitigates the level of *buffer occupancy* of path routers, which is desirable because the path routers might not want to hold transit packets for a long time. That is, it will not result in large buffer buildups, because of the per-node basis end-to-end bandwidth estimation as follows.

3.4.3 The instantaneous bottleneck throughput

Figure 3-6 illustrates that in the framework of 802.11 MAC protocol, the total amount of time taken for successful transmission of a datagram incoming (for example, IP datagram 7 on the top of router A' IFQ) signifies the *MAC related delay* including the total amount of time to dequeue packets arrived in advance (i.e., $T_3 + T_4 + T_5 + T_6$ as a FIFO manner).

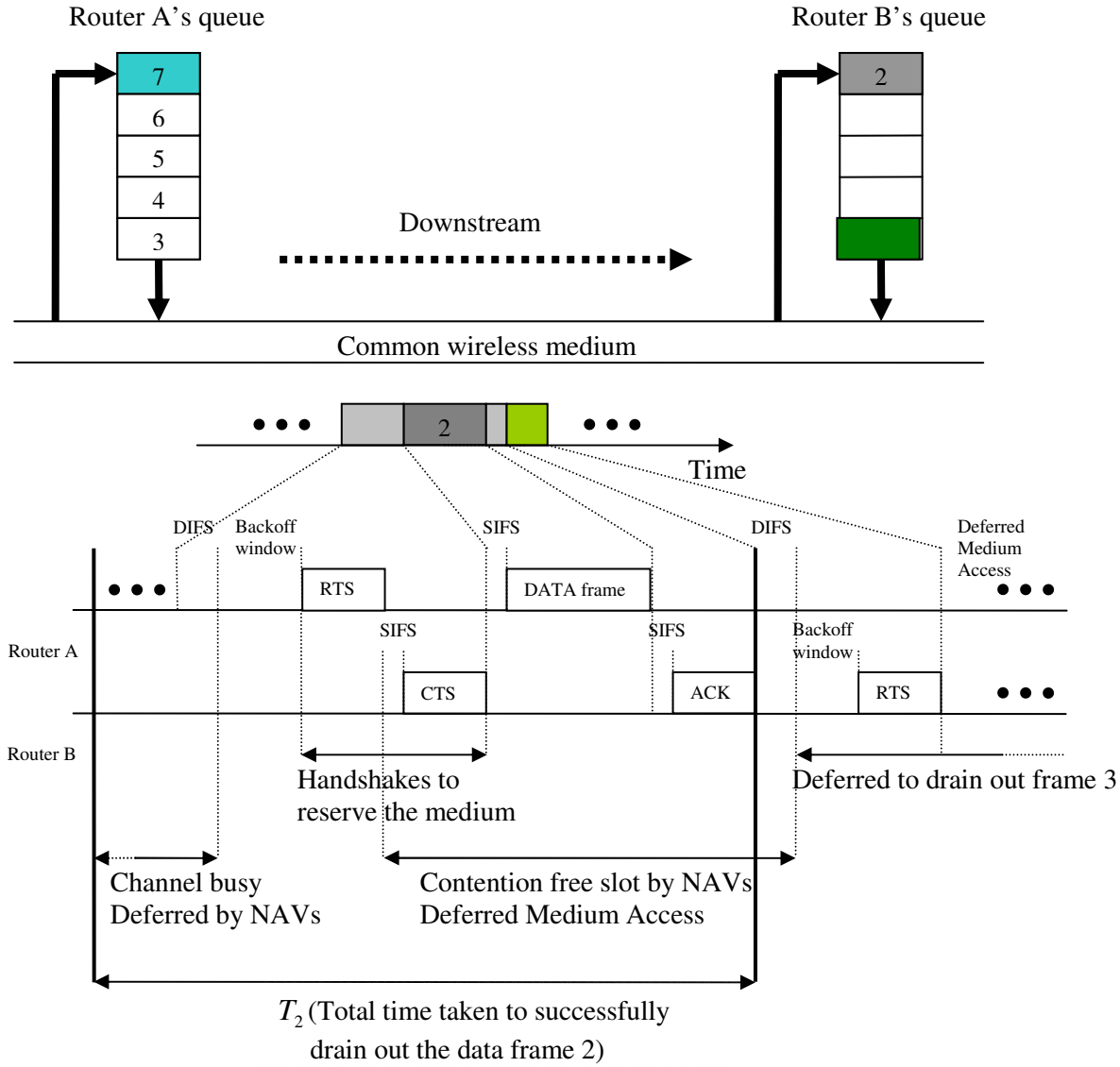


Figure 3-6. Delay contributions in forwarding packets through a router in a simplicity manner

Each packet transmission requires time for MAC related 4-way handshake accordingly as well as the medium access deferred time by NAVs and backoffs. Time taken per packet varies by those like—Early RTS-CTS exchange failures requiring additional transmission coordination time, and

failures in transmission of data frames[1] caused by either the effect of the hidden terminal problem (at extremely high mobility causing no NAV set properly) or wireless channel errors[2].

As a result, illustrated from Figure 3-6 and Figure 3-7, it shows time taken to successfully drain out a packet at the head of the queue in the router A, which gives bytes/sec equal to the instantaneous throughput as shown in Equation 3-1.

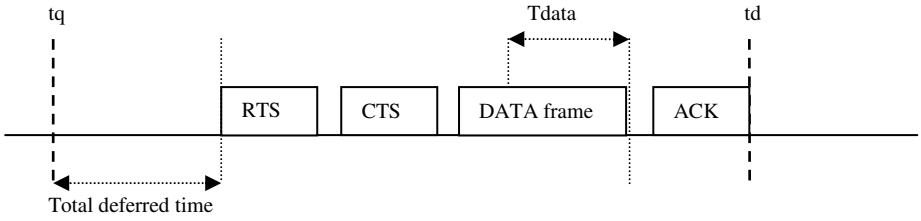


Figure 3-7. Instantaneous throughout measured at a router [42], where tq = the time queued, td= time dequeued, and Tdata = P/B_{raw} , time for transmitting a TCP packet .

$$S_{instantaneous} = \frac{P}{t_d - t_q} = \frac{P}{T_3 + T_4 + T_5 + T_6 + T_7}$$
$$= \frac{P}{queuing\_delay + RTS/CTS/DATA_7/ACK}$$
$$= \frac{P}{MAC\_related\_delay + P/B_{raw}}$$

Equation 3-1

Where T_7 [3] is the delay taken to successfully relay the data frame 7, denoted as $RTS/CTS/DATA_7/ACK$.

[1] Each data frame incoming or outgoing is packetized to include PHY Header, MAC header, TCP/IP header and TCP payload (i.e., MSS), or no TCP payload if it is an IP related control datagram like ICMP or routing control packet.

[2] In case of high mobility of high density nodes, higher BER could also yield considerable high MAC contention-like delay, such as, RTS-CTS failures due to higher BER, and MAC ACK reception error due to sudden contending intruders. The wireless high BER related delay will be thus included in the MAC related delay because of no way of confident distinction and no need between contention induced and BER induced losses in such dynamic circumstances. In fact, the primitive CSMA/CA mechanism is not able to distinguish losses among the other end moving out of transmission range (route disconnection), high contention, and wireless channel corruption. In any reason, just after a finite number of times, the MAC layer concludes a link failure and informs the higher layers. Most routing protocols for use in ad hoc networks, such as DSR and AODV, are using such a MAC feedback to trigger route failure message and invoke the route rediscovery procedure.

[3] $T_7 = (Deferred\_periods + DIFS + (Back\_off + RTS + \sigma + SIFS + CTS + \sigma + ?) +$
$SIFS + DATA_7 + \sigma + SIFS + ACK + \sigma + ?)$

Where σ is propagation delay, "?" means additionally time required meaning that any of preceding frames has been lost or not received within a certain time interval.

3.4.3.1 The Impact of packet size variation

As known in Equation 3-1, as a matter of fact, each instantaneous link throughput measured is varied by a packet size, as seen in Equation 3-3 (because the necessary lower overheads are equally required each time a packet is transmitted over irrespective of the packet size).

Once the packet size varies at the sender due to being restricted by path MTU just in case, the instant per-node delay may be approximated like $P'/S_{p'}$. Thus, the sender should adjust its congestion window in multiple units of the varied packet size accordingly in the sense that a larger MSS requires more time to traverse the path than previous. But, more importantly, the sender also wants to control the fluctuating bandwidth portion related to MAC overheads rather than that related to the larger but fixed MSS by itself—for example, [63] has shown that there exists an optimum window size (best throughput at a window size of 3, experimented in a 7-hop topology with three different packet sizes, 576, 1024, and 1460 bytes), irrespective of packet size, meaning that it requires a certain (consistent) amount of MAC overheads. Thus, supposing a certain fixed amount of MAC overheads per a transmitting packet, the sender can vary the size of transmitting packets (i.e., varying MSS because of a fixed default TCP/IP header). Equation 3-2 determines a maximum possible throughput as a function of the varied packet size and accordingly gives an adjusted congestion window. On the other hand, as further addressed in Chapter 3.4.7.2, the receiver can also facilitate it in order to rectify the *congestion window delimiter* in change of incoming packet size.

$$S_{p'} \cong \frac{P'}{MAC\_related\_delay + P'/B_{raw}} = \frac{P'}{\left(\frac{1}{S_p} \times P - \frac{P}{B_{raw}}\right) + \frac{P'}{B_{raw}}} = \frac{S_p \times P' \times B_{raw}}{P \times B_{raw} + S_P(P'-P)}$$

Equation 3-2

Thus,

$$\frac{P'}{S_{p'}} = \frac{P \times B_{raw} + S_P(P'-P)}{S_P \times B_{raw}} = \frac{P}{S_P} + \frac{(P'-P)}{B_{raw}} \quad \text{as per-hop inflation}$$

Equation 3-3

Where P' is the varied packet size, P is the previous size, and $S_{p'}$ is the throughput with the varied P'.

In case, varying to a larger MSS may suffer from RTOs (because the larger segment, the higher transmission delay and thus the larger RTT). As a result, the number of in-flight packets is growing slower and in turn there might be insufficient number of DUPACKs in time to trigger a fast retransmit. Defining a specific MSS with an upper bound the path MTU discovery imposes is, however, out of the scope of this thesis. A fixed sized MSS will be just used. The set of the "Don't fragment" bit[1] at sender will make the byte-stream based advertisement behave as the packet-stream based advertisement to control the frequency of the sender's medium access.

Facilitating the *packet stream-like byte stream based advertisement* informs of maximum upper bound network capacity specified by a certain packet size observed at the receiver. Therefore, it justifies, at the sender the use of the path MTU discovery to have a fixed segment size (i.e., a certain MSS with an upper limit that the path MTU determines, RFC1191 [103] unless required to be reduced) and, at the receiver the byte stream advertisement to be equivalent to the multiple units of the MSS.

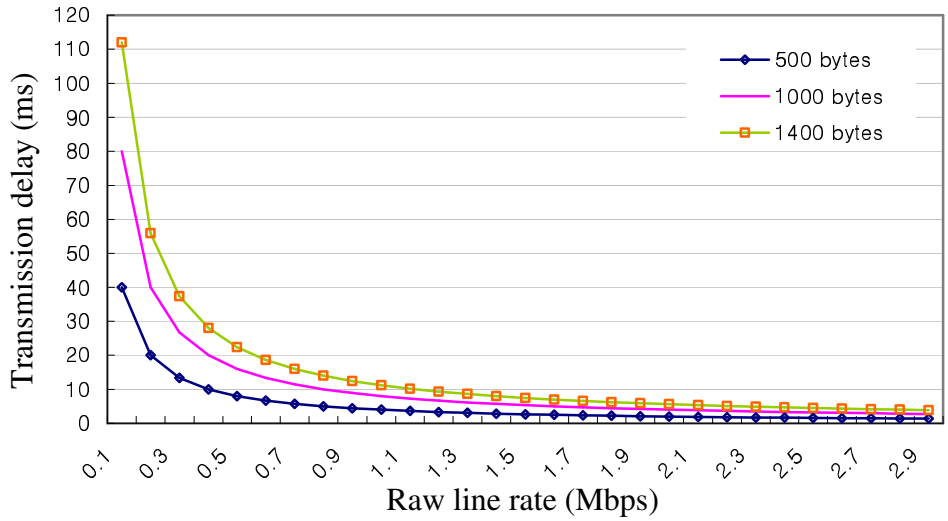


Figure 3-8. Per hop transmission delay for different packet sizes and raw line rates. Transmission delay per hop is not much distinctive for different packet sizes, but over multi-hops, subsequently inflated RTT and wireless BER become problematic.

[1] MAC layer frame fragmentations, due to size discrepancy between layers, or other reasons such as high BER, will not be aware of in TCP layer. Such induced delay factor will be considered as comprehensive MAC related delay. In this study, the use of PMTU (i.e., the use of TCP MSS option with the DF bit set) was assumed to expose all the network condition changes, such as wireless error, routng failure, and buffer congestion.

3.4.4 Forward Link Delay Variation (FLDV)

Proposals introduced in ADTCP-friendly [58] and TCP santa cruz [112] used a metric, inter-packet delay[1] measured at receiver-end and sender-end, respectively, in support of the timestamp option, between two consecutively arriving packets.

The use of the inter-packet delay difference is facilitated to distinguish between wireless link error and congestion loss, for example, [28, 29] for hybrid (wireline and wireless) links, [112] for wireline link of path asymmetry, and [58] for multihop wireless links.

However, it may no longer be a meaningful representative by itself when accompanied with reliable link level schemes because, in either case of congestive buffer or error-prone link, substantial inter-packet delay difference might be observed between packets. In this sense, in term of controlling the transmission rate against instantaneously changing network condition, we need not distinguish the two disparate cases because, if the inter-packet delay difference is enough high to be recognized, it is more reasonable that the receiver needs to limit the sender anyway. Afterwards, such circumstances resolves, the rate will be again restored as fast as possible.

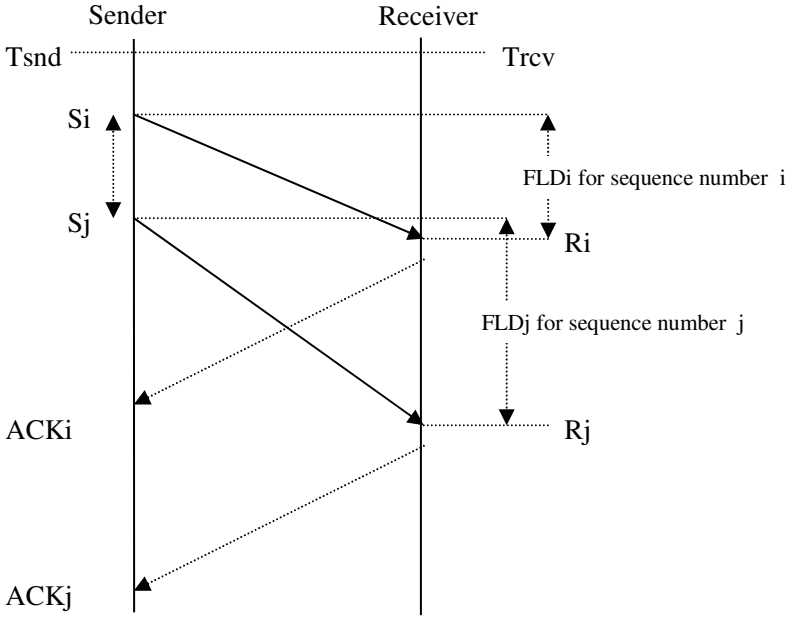
3.4.4.1 Use of FLDV

The variation of forward link delays between any two packets is called Forward Link Delay Variation (FLDV).

As shown in Figure 3-9, it is computed by any pair of packet observed at the receiver in a chronicle order. Comprehensively, it implies the forward link condition of network path and importantly is robust to the cases of path asymmetries because an ad hoc multipath routing may be utilized.

The receiver-oriented flow rate controller will use forward link delay (FLD) and its variation (FLDV), throughout packet-by-packet scrutiny, meaningfully representing the fluctuation of link condition and particularly identifying the dynamics of medium availability, which is probably imposed by NAVs, back offs, local retransmissions, and route disconnection.

[1] As proposed in TCP-santa cruz [112], this metric was used for implementing a congestion control algorithm based not upon the ACK clocking in general but the delay variation of forward link delay, which is robust to ACK losses.



$$FLD_k = R_k - S_k \qquad \textbf{Equation 3-4}$$

$$FLDV_{j.i} = FLD_j - FLD_i \qquad \textbf{Equation 3-5}$$

Where, *i, j*, and *k* are sequence numbers

Figure 3-9. Measurement of the forward link delay variations

Equation 3-4 shows the FLD experienced by packet k, but it means a connection-specific and not so meaningful forward link delay unless the clocks of both ends are synchronized between end hosts (i.e., different initial times and skew factors characterizing identical time and increment, for which the Network Time Protocol (NTP) [102] is widely used in the Internet for clock synchronization, and it provides accuracy of the order of milliseconds under reasonable circumstances, and [105] introduces the algorithm that is simple, fast and robust in the clock synchronization in terms of both the clock offset[1] and in particular the skew[2]). In this theis, in order to validate the FLD, the clocks in between are supposed to be synchronously ticking. Appendix A.1 presents a simple passive method in order to synchronize the clocks between end hosts. Details are put in a further research.

[1] The difference between the time of a clock and the true time
[2] The difference in the ticking frequencies of a clock and the true clock

Supposed in convenience that the clocks are identically ticking, the receiver can keep monitoring the FLDs and using that, will alter the rate accordingly each time data packet arrives. Beneficially by use of timestamp of each packet, the FLD can be renewed by each receiving packet recently originated in time, regardless of packet sequence numbers. Thus the rate computed in view of FLD being sampled will be altered independently of the packet sequence numbers. Even though packet arrives but is out of sequence (that is, from Equation 3-5, $(j-i)$ *is* equal to or greater than 2 (i.e., $j > i+1$)), the receiver just keeps sampling if it has not a stale timestamp, and keeps limiting the sender (rather than tries to identify the cause of the packet absence as in ADTCP-Friendly [58]). In the meantime, the sender will make an appropriate decision in case of packet absence.

3.4.5 Acceptable maxima and minima of each hop link delay

Unlike other wireline networks, in ad hoc networks, primary reason causing the unpredictable (likely fluctuating due to the density of competing nodes) delay experienced by packets, is due to the characteristics of MAC protocol employed. The employed 802.11 MAC protocol plays an important role in sharing the common channel for all competing nodes in a fair manner. Thus, it is a major factor to determine the bandwidth availability, which may be available instantaneously (i.e., inter-dependency of each other) due to high mobile nature of nodes and unpredictable, transient interventions (short bursts).

Envisaging such instantaneousness, the followings give an intuition that instantaneous throughput determined by forward link delay observed by a traveling packet should be delimited between extreme values determined by path identity, such as the number of hops, packet size, raw line rate, and others. They can validate instantaneously fluctuating estimated bandwidth because boundary levels of one hop delay (indicating packet service time per-node) can permit either most or least change of throughput (i.e., S_{max} and S_{min})

3.4.5.1 Bandwidth fluctuation due to the dynamic MAC nature

Figure 3-10 shows the instantaneous throughout of each hop link, through which a packet traverses and then experiences considerable delays accordingly. In ATP [139] and TCP-EXACT [42], intermediate routers inform the sender of available bandwidth whenever a packet transits through, based on per-node and per-flow computations, respectively.

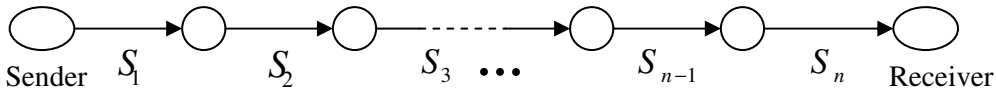


Figure 3-10. Hop based available throughput (byte/sec) (*n* hops)—each available throughput, S, means an instantaneous channel capacity when a packet outgoes through a router and implies per-packet delay experienced by a packet at each intermediate router it traversed through, fluctuated by the level of contention, mobility, and queuing.

In 802.11, traffic stress on a router along the path causes contention and ends up with the degradation of available throughput fluctuating adjacent neighbors' throughputs due to common channel sharing mechanism. As Equation 3-1 implies, hop link throughput available at a node influences the medium availability of other competing nodes (one or two hops apart, in a well-positioned string chain as in Figure 2-5 in Chapter 2.3.4) around the node. On the other hand, even though no such competitors are present and so a single transmitting node is able to occupy the medium, the bandwidth may not be available due to other preceding packets queued for others (related to buffer queuing) or to local retransmissions of packets by itself (caused by multiple wireless corruptions).

As a result, the dynamic interdependency of mobile nodes confirms that resultant MAC related delay dynamically fluctuates over time (with regards to the ubiquitous nature of mobile nodes that causes fluctuating degree of contention among the nodes). Thus, each hop link throughput is likely to be a transient throughput and so validated only instantaneously.

In terms of the computation of each hop link throughput at each router, the least available capacity as a bottleneck is a key so as to limit the sender's flow rate, but presumably the rate informed from the bottleneck router is likely to be out of date due to long links traversed. Thus, such router involvements might not even attain reasonable improvement because of no guarantee of timely delivery of valid information, even though it takes massive effort to implement.

3.4.5.2 Microscopic view of n-hop link Forward Link Delay (FLD)

Given that Figure 3-10 shows an *n*-hop link inside the network shown in Figure 3-5 between source-destination pair, the instantly measured available throughput of each hop link is denoted as $S_1, S_2, \ldots, S_n$. Each determines time taken to forward a packet at an instant over each link, namely,

the *instantaneous per-node delay*—i.e., the amount of time spent for servicing a packet[1] in a router (per-packet delay), mostly affected by neighboring medium competitors and pre-built queue.

Actual forward link delay (FLD) experienced by a packet, P, passing through the *n*- hop link:

$$FLD = \frac{P}{S_1} + \frac{P}{S_2} + ... + \frac{P}{S_n} \qquad \textbf{Equation 3-6}$$

Given that,

$$\text{Maximum throughput } (\mathrm{S}_{\max}) = \max(\mathrm{S}_{1,} S_2, ..., S_{n-1}, S_n)$$

$$\text{Minimum throughput } (\mathrm{S}_{\min}) = \min(\mathrm{S}_{1,} S_2, ..., S_{n-1}, S_n)$$

With packet size in byte, P, the total time of the delays taken by a packet traversing through the *n* hop link varies due to changes in each hop link capacity, each of which lies between the two extreme values determined by $\mathrm{S}_{\max}$ and $\mathrm{S}_{\min}$ representing the delay through the best link (i.e., throughput over the best link) and the worst link (i.e., bottleneck capacity but no breakage) respectively, as the followings:

$$n\frac{P}{B_{raw}} < \left(\frac{P}{S_{\max}} + ... + \frac{P}{S_{\max}} = n\frac{P}{S_{\max}} \right) \le \frac{P}{S_1} + \frac{P}{S_2} + ... + \frac{P}{S_n} \le \left(\frac{P}{S_{\min}} + ... + \frac{P}{S_{\min}} = n\frac{P}{S_{\min}} \right)$$

Equation 3-7

Where, nP / B_{raw} identifies fundamental transmission delay per packet over *n* hops characterized by a line rate B_{raw} and a packet size (i.e., traversing time taken for a pure TCP segment).

With respect to *reference metrics* for a specific path link, we use these two extremes ($\mathrm{S}_{\max}$ and $\mathrm{S}_{\min}$) to determine the path identify, so both extremes are used as deterministic metrics to explain an allowable range of the bottleneck throughput that varies. For a given path, $\mathrm{S}_{\max}$ represents the best link condition without MAC collision and queuing delay, only taking into account lower layer overheads per packet (e.g., determined by a minimum delay obtained by one packet traversing through a stationary path without other interfering nodes, where $nP / S_{\max}$ represents the delay

[1] The term used is equivalent to the maximum average delay (*avg(D)*) (average transmission and queuing delays) as addressed in [139].

taken for successful RTS/CTS/DATA/ACK handshakes over the *n*-hop link). S_{min} means the worst link throughout experiencing the extreme MAC related delay (long back offs, NAVs, and link layer retransmissions due to corruptions, which all causes subsequent queuing delay).

Rewriting the FLD: $FLD = \frac{P}{S_1} + \frac{P}{S_2} + ... + \frac{P}{S_n}$

$$FLD = total\_MAC\_related\_delay + \left(total\_transmission\_delay = n\frac{P}{B_{raw}} \right)$$

Equation 3-8

Thereby,

$$n\frac{P}{B_{raw}} < n\frac{P}{S_{\max}} \le \left(\left(n \times average\_MAC\_delay + n\frac{P}{B_{raw}} \right) = n\frac{P}{S_{average}} \right) \le n\frac{P}{S_{bottleneck}} \le n\frac{P}{S_{\min}}$$

Where $total\_MAC\_related\_delay = n \times average\_MAC\_delay$

Equation 3-9

As shown in Equation 3-9, the simplified FLD of Equation 3-8 is ranged between the extreme delays determined by S_{max} and S_{min}. The best and the worst boundary link bandwidth validate a range that the link capacity can vary. $S_{a\,\mathrm{var}\,age}$ means the average throughput of each link and further explains the maximally allowable bottleneck throughput, which is used as an initial point of an optimum rate. The receiver converges it to a bottleneck throughput, accounting for subsequently arriving packets.

3.4.6 Determination of *Sbottleneck*

As in Figure 3-10 identifying per-node capacity, each instantaneous link throughput determines instantaneous per-node delay, each of which is bounded between two extreme delays as seen in Figure 3-11.

In Figure 3-11, at a router where a packet experiences the longest per-node delay, denoted as $P/S_{bottleneck}$, the link determines the most dominating bottleneck throughput.

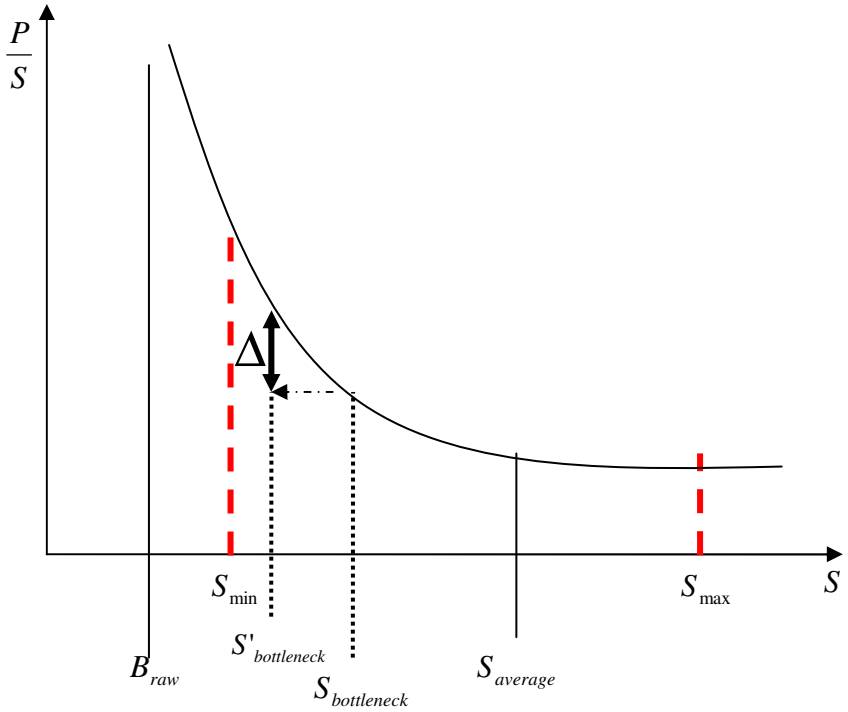


Figure 3-11. Per-node delay experienced by a traversing packet

Thus, keeping under sending rate allowed by $S_{bottleneck}$ promises that the sending rate does not overwhelm the available network capacity. Since per-node delay is time-variant and gets worse or better, the resultant bottleneck throughput ($S'_{bottleneck}$) is going to be:

$$S'_{bottleneck} = \frac{P}{\frac{P}{S_{bottleneck}} + \Delta} \qquad \textbf{Equation 3-10}$$

Where Δ means the delay variation observed at the bottleneck when more stress is applied as in Figure 3-11.

As in Equation 3-10, the bottleneck capacity changes according to the delay observed in the bottleneck router. Given the same propagation delay because the packet traverse through the same route to the receiver and the same transmission delay because of the same packet size in use, the variation of FLD thus represents variable queuing delay, which is then meant to be the variable *MAC related delay* (resulting from bandwidth-limited ad hoc link characteristics by 802.11 MAC protocol).

Factors why the delay contributed by the hop link has been induced and fluctuated at the router are specifically summarized:

1. Neighboring concurrent competitors (put into NAVs),
2. Wireless link corruptions taking up a number of local retransmissions,
3. Route breakage requiring additional delay for restoring the link or absorbing unsuccessful local retransmission(s), and
4. Transmissions or receptions of other flows (i.e., serving for others).

The rate computed considers the above four factors into account; however, in case, the receiver might not, or should not, have to take into account the delay absorbed by the factor 4, which is the queuing delay—for example, in the connection with a wireline opponent suffering from built-in buffers, the receiver cannot alleviate the buffer queuing problem in the wireline network because it is not dominated as in wireless link capacity. To eliminate the impact of the above factor 4 to a certain degree, the rate computed takes in effect only when the deterministic *buffer occupancy* (Chapter 5.1.4) exceeds a certain threshold.

3.4.6.1 Bottleneck throughput ranging from $S_{\min}$ to $S_{a\,\mathrm{var}\,age}$

Figure 3-11 illustrates the delay contribution of each hop link when a packet traverses through each hop link. $S_{bottleneck}$ lies between $S_{\min}$ and $S_{a\,\mathrm{var}\,age}$ for the least and the most, respectively, and $S_{a\,\mathrm{var}\,age}$ is less than, or equal to, $S_{\max}$.

Thereby, simply rearranging Equation 3-9,

$$S_{\min} \le S_{bottleneck} \le S_{average} \le S_{\max} \qquad \textbf{Equation 3-11}$$

$S_{\max}$ defines a maximally allowable hop link capacity in multi-hop ad hoc links, where $P/S_{\max} = RTS/CTS/(P/B_{raw})/ACK$ as a time taken for the 4-way handshake of a packet, P, and thus $S_{\max}$ will be identified by a maximally allowable throughput as a path-specific *reference metric*. This entity in particular delimits $S_{average}$, which is as a function of FLD and current hop

length. If $S_{average}$ exceeds $S_{\max}$, it is meant to be likely that the receiver has been communicating with a wireline counterpart ($S_{average}$ might exceed $S_{\max}$ because wireline hop link[1] throughput is much higher than ad hoc wireless link one), or necessary minimum MAC overheads has been reduced somehow. If instantaneously measured $S_{average}$ is greater than $S_{\max}$, the receiver could thus suppose the presence of faster link(s) between end correspondents. Furthermore, $S_{average}$ could be facilitated as a metric for admission control of a real-time transport protocol because this metric is supposed to determine a hop-based capacity, along the path, independent of RTT between flows (i.e., unfairness might be caused due to difference in RTT between long and short flows).

$S_{\min}$ defines a minimally required hop link capacity. If the receiver detects that the rate adjusted goes below $S_{\min}$, it might expect that the current hop link is going to be (or seems to be) disconnected because too low throughput implicitly means the consequence of highly induced MAC related delay, and thus jitter effect in arrival times (data or ACK starving in either forward or reverse way). Nominally, given that $S_{\min}$ is greater than $\frac{P}{FLD-(n-1)P/S_{\max}}$ with the longest FLD, we use other value to symbolize $S_{\min}$, thus to determine the likelihood of route breakage because in fact $S_{\min}$ itself can not be directly related to assess an imminent route breakage. In terms of bandwidth constraint 802.11 MAC characteristics, the freezing timer (Chapter 3.4.8) can determine $S_{\min}$ (i.e., the reciprocal of the freezing timer multiplied by MSS) because allowable traveling delay prior to TCP sender timeout could specify $S_{\min}$ in the sense that the freezing timer determines the least link capacity in terms of falsely induced RTOs at the TCP sender.

With assumption of synchronized clocks between end hosts, the initial bottleneck throughput is defined as, started from, the initial average, $S_{average}(0)$ of the initial FLD, each time a connection is established or the route is switched.

$$S_{bottleneck}(0) = S_{average}(0) = \frac{nP}{FLD_0} \qquad \textbf{Equation 3-12}$$

Where, P is a segment (i.e., MSS) that does not include TCP/IP headers (the congestion window is multiple units of MSS and so the receiver feedbacks the advertised window in unit of the MSS.

[1] i.e., faster line rate and much less suffering from medium access, given that a reasonable buffer exists.

Then, forward link delay variation will evolve bottleneck capacity subsequently

$$S_{bottleneck}(1) = \frac{P}{\frac{P}{S_{bottleneck}(0)} + \Delta_0} = \frac{P}{\frac{FLD_0}{n} + \Delta_0}$$

Thus, the normalized form is:

$$S_{bottleneck}(i+1) = \frac{P}{\frac{P}{S_{bottleneck}(i)} + \Delta_i}$$

Where $S_{bottleneck}(0) = nP / FLD_0$. Each time *n* changes, *i* goes to 0.

The delay (Δ) is determined by use of FLDV.

$$S_{bottleneck}(i+1) = \frac{P}{\frac{P}{S_{bottleneck}(i)} + \Delta} = \frac{P}{\frac{P}{S_{bottleneck}(i)} + \delta \times FLDV_{i+1,i}}$$ **Equation 3-13**

Where, δ is a scale factor, *1/n* in use, that may vary for fast convergence, and $S_{bottleneck}(0) = nP / FLD_0$. Each time *n* changes, *i* goes to 0.

As a matter of fact, $S_{bottleneck}(0)$ coarsely delimits the congestion window by a maximally allowable bound bottleneck capacity. In addition, a wrong initial capacity due to clocks not synchronized could be derived. But, the rate will converge to an optimum value sooner or later by subsequently receiving packets. The reasoning is because each receiving packet is sampled by variations of FLD (FLDV), which is independent of the clock synchronization (however, without a doubt, unequal skew rate may be problematic).

3.4.6.1.1 The use of δ

A packet in a specific size originated from the sender will traverse through a certain lengthy route, experiencing a certain amount of delays. Basically, packet-by-packet FLDV scales fluctuating level distinctively according to the hop length applied as well as specific lower layers operation that might additionally triggers control packets. The receiver should have a scaling factor to keep validating the degree of medium contention, by which it can account for a more accurate rate.

3.4.6.1.2 Uncertainty of the scale factor of δ

From Equation 3-11, the term, $\delta \times FLDV$, represents an additional, or subtractive, delay absorbed by a bottleneck router. Equation 3-9 and Equation 3-11 gives, for bottleneck capacity in response to *(i+1)th* packet, the least and the most value of δ as following:

$$S_{bottleneck}(i+1) = \frac{P}{\frac{P}{S_{bottleneck}(i)} + \Delta} \le \frac{P}{\frac{FLD(i+1)}{n}} = S_{average}(i+1)$$

Therefore,

$$\Delta = \delta \times FLDV_{i+1,i} \ge \frac{FLD(i+1)}{n} - \frac{P}{S_{bottleneck}(i)}$$

In turn,

$$\delta \ge \frac{\frac{FLD(i+1)}{n} - \frac{P}{S_{bottleneck}(i)}}{\left| FLDV_{i+1,i} \right|}$$

From the above, since *i* is 0, δ approximates to *1/n*, as the least at positive FLDVs and as the most at negative FLDVs. Meanwhile $S_{bottleneck}$ must be greater than, or equal to, $S_{\min}$ (if defined), it gives the other bound value of δ accordingly.

In the simulator, if we use this value, the bottleneck throughput becomes equivalent to $S_{average}$. Because of ambiguity of δ, of which the receiver is not aware (taking the coarse term, *1/n*), the estimated rate must be strongly jittering and so need moving averaging as in Equation 3-14. (For the study, we used the term, 7/8 for the moving average, see Equation 3-18)

The bottleneck will be in use as follows:

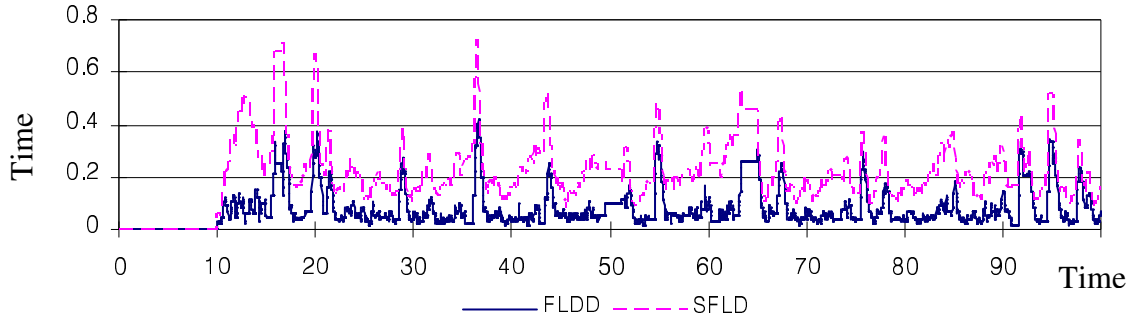
$$S_{bottleneck}(i+1) = \frac{P}{\frac{P}{S_{bottleneck}(i)} + \frac{1}{n} FLDV} = \frac{P}{\frac{FLD(i+1)}{n}} = S_{average}(i+1)$$

Then,

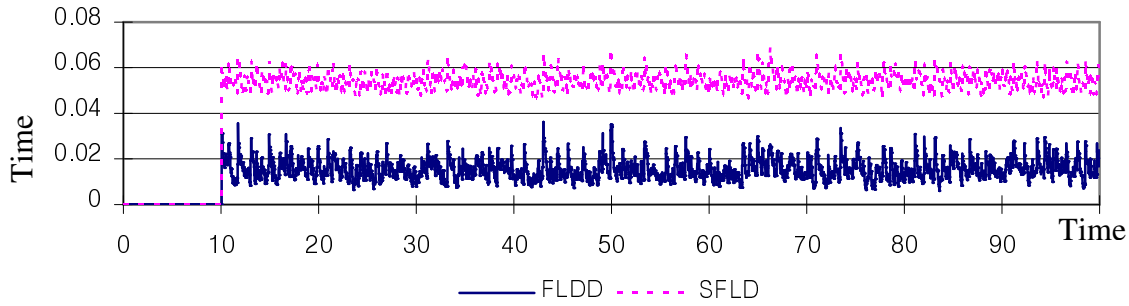
$$S_{bottleneck} = \frac{nP}{FLD} \approx \frac{nP}{SFLD} \quad \because \text{to alleviate the rate jittering}$$

Equation 3-14

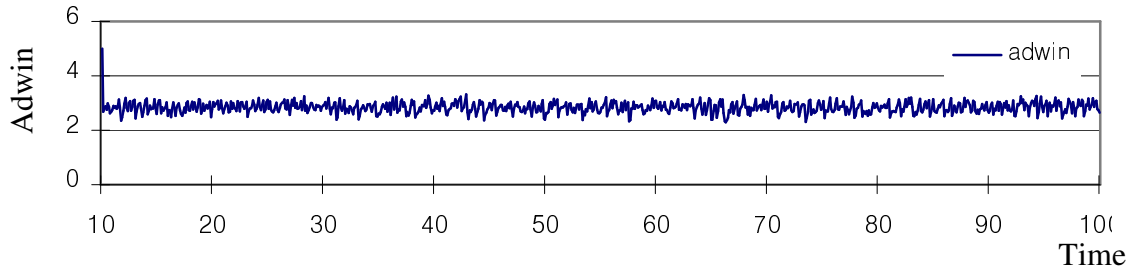
Once we apply, namely, the receiver's *rater* of Equation 3-14 taking a fixed *1/n* for δ, it will regulate FLD, as verified below in Figure 3-12 (b), to flat rather than estimate the available throughput because of the fixed δ applied, which does not impose the exact varying portion (Δ) of delay observed at the bottleneck.



(a) Solely congestion window limited. The nature of the baseline generally results in high network buffer utilization, causing large RTT and thus, large RTOs—long timeouts and thus, reduced TCP throughput.



(a) Obtained by the explicit window derived from the *rater* with *1/n* of scale factor, about 10 times less than (a)



(b) Advertised window driven from the *rater* multiplied by FLDmin, fluctuating 2 to 3 as converged.

Figure 3-12. FLDD and SFLD between receivers with and without the rater. It gives its advertised window under control (5 hop stationary string path of a bulk FTP transfer and packet size 500 bytes, active 10 to 100 sec). The use of packet size 500 bytes also helps with the fast delivery of packets, compared to 1000 bytes.

In other words, other heavy traffic loads applied may cease transmission because of the *rater*'s converging close to zero. In case, if some of path routers suffer from heavy buffer congestion, the ad hoc receiver almost always advertises zero receiver's window because we use the *comprehensive MAC related delay* that includes the queuing delay. Therefore, the advertised explicit window should be bounded between two extremes, the lower bound of which gives sustainable but guarantee a minimum transfer in terms of steady flow of the receiver's feedback, and upper of which defines a maximum expansion of window the rater can evolve.

3.4.7 Explicit window advertisement

This Section is subject to a certain link condition that as addressed in Chapter 2.3.4, IEEE 802.11 characterizes specifically, such as valid transmission and interfering ranges, and primitive half duplex nature of the link (given that every node is well positioned to communicate with each other up to one-hop distant neighbors and to interfere up to two hop distant vicinities). Then, the lower and the upper bounds delimiting the receiver's advertised window, so-called *congestion window delimiter* (*delimiter*, for short), will be assessed.

Because TCP basically uses the window based congestion control mechanism, the estimated rate is multiplied by a RTT in order to compute the *bandwidth-delay product*. Accordingly, the sender passes down the available number of packets into the network *pipeline* for a RTT-euqivalent time. In practice, we may use RTT obtained by twice of SFLD, or of minimum FLD ever tracked.

The initial receiver's advertised window (*adwin*) can be therefore as below:

$$Adwin_0 \cong \frac{S_{average}}{P} \times RTT_{\min} = \frac{n}{FLD_0} \times RTT_{\min} \cong \frac{n}{FLD_0} \times 2 \times FLD_0$$
$$= 2n \qquad (segments)$$

The initial *adwin,* valued *2n,* implies that we compute the rate by the per-node based packet forwarding time and that the sender puts *2n* packets in the network *pipeline* initially because of the equal share of RTT over *2n* hops each packet traverses through—i.e., as burst transfers, each hop ideally should at most have one packet forward and at the same time one another reverse; otherwise, queue builds up and RTT increases. However, due to the limitation of, half duplex, 802.11 MAC protocol that uses a single common channel to share, so that each link can only transmit one packet at a time, the network pipeline cannot hold *2n* packets any more to relay concurrently, but possibly

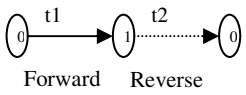
n packets at most. As early mentioned in Chapter 2.3.4.1 about MAC collision problems, neighboring nodes even in well-distant string topology, where one node can only transmit to one-hop away node and might interfere up to two-hop distant nodes, interrupt the packet transmission of at least its two-hop distant neighboring nodes. As a result, *n* packets are not likely to be relayed concurrently.

Mathematical approaches are found in [73] to evaluate the allowable utmost explicit windows according to the number of hops between end nodes. However, the analysis is evaluated under the assumption that the forward link is identical to the reverse link. The following shows an optimally maximally allowable number of packets supposing no collisions in a given hop length, taking into account multi-path routing (it only cares of the forward path link). According to Chapters 3.4.3, 3.4.5.2, and 3.4.6, per-node throughput is obtained by per-packet service time (i.e., queuing and transmission delays experienced by every outgoing packet at intermediate router) over each hop link.

The following evaluates a possible maximum amount[1] of transit packets along a certain hop length, in order not to suffer from any MAC collision. Suppose each hop is just within the transmission range of its one-hop neighbors but out of that of two hop distant neighbors (as in Figure 2-5 in Chapter 2.3.4).

---

1 hop:



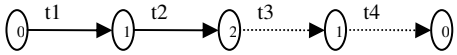
Where, t1, t2, … is the hop link delay and determines the instantaneous hop link throughput.

$$W_{\max} = bandwidth \times delay = \frac{1}{t1} \times t1 + \frac{1}{t2} \times 0 = 1 \qquad packet$$

Thus, $W = \frac{1}{t1} \times FLD$ (Effective throughput = $\frac{1}{t1+t2}, pkt/\sec$)

[1] Dimensioning the network *pipeline* specifically characterized by 802.11 MAC protocol.

2 hops:



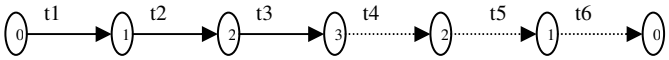
$$W_{\max} = MAX\left(\sum_{i}^{n} hop\_link\_throughput_i \times hop\_link\_delay_i\right)$$

Suppose $t1 > t2$,

$$= \frac{1}{t1}\times t1 + \frac{1}{t2}\times 0 + \frac{1}{t3}\times 0 + \frac{1}{t4}\times 0 = 1 \quad packet$$

Thus, $W = \frac{1}{t1}\times t1 \geq \frac{1}{t1}\times\frac{FLD}{2}$ (Effective throughput = $\frac{1}{t1+t2+t3+t4}, pkt/\sec$)

3 hops:



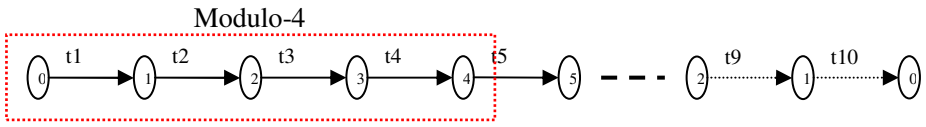
Suppose $t1$ = max (t1, t2, t3, t4, t5, t6),

$$W_{\max} = \frac{1}{t1}\times t1 + \frac{1}{t2}\times 0 + \frac{1}{t3}\times 0 + \frac{1}{t4}\times 0 + \frac{1}{t5}\times 0 + \frac{1}{t6}\times 0 = 1 \quad packet$$

Thus, $W \geq \frac{1}{t1}\times\frac{FLD}{3}$ (Effective throughput = $\frac{1}{RTT}, pkt/\sec$)

⋮

5 hops/General form:



Suppose $t1$ = max (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10),

$W_{\max} = \frac{1}{t1}\times t1 + \frac{1}{t5}\times t1 \geq 2$ *packet s*, this inequality means more than 2 but up to t1/t5,

(in a general form, $Int\left(\frac{P/S_{bottleneck}}{P/S_{\max}}\right)$ *packets* at most, now referred to as an *offset*,

where Int(x) is the round-down of x).

Thus, $W \geq \frac{1}{t1}\times\frac{2\times FLD}{5}$ (Effective throughput $\geq \frac{2}{RTT}, pkt/\sec$)

At least, more than, or equal to, two packets are incremented every modulo-4 cycle.

The offset derived, taking into account the modulo-4 cycles, varies up to $Int\left(\frac{S_{\max}}{S_{bottleneck}}\times(\mathrm{mod}\,ulo4-1)\right)$ *packet s*, and might be accounted to each optimum window of

Equation 3-15 in order to determine the upper bound of the optimum window. Imposing the upper bound of the optimum window (i.e., max *offset*): (however, it cannot assure of its presence of the offset due to MAC deficiency, the lack of ideal packet scheduling to transmit.)

$$W_{optimum} \le \mathrm{mod}\,ulo4 + Int\left(\frac{S_{\max}}{S_{bottleneck}} \times (\mathrm{mod}\,ulo4 - 1)\right)$$

Equation 3-15

Where, practically $S_{\max} \approx \dfrac{P}{\min\_ forward\_hop\_delay}$

3.4.7.1 Evaluation of $W_{optimum}$ in 802.11 MAC protocol

As figured out in Chapter 3.4.7, the conservative explicit advertised window will be as following: (as a general form of the optimum window)

```
modulo4 = int(n/4); // round down
If (n % 4 != 0),
        ++Modulo4;
```

Then,

$$W_{optimum} \le \frac{S_{bottleneck}}{P} \times \frac{\mathrm{mod}\,ulo4 \times FLD}{n} \approx \mathrm{mod}\,ulo4 \quad \left(= \frac{S_{average}}{P} \times \frac{\mathrm{mod}\,ulo4 \times FLD}{n}\right);$$

Equation 3-16

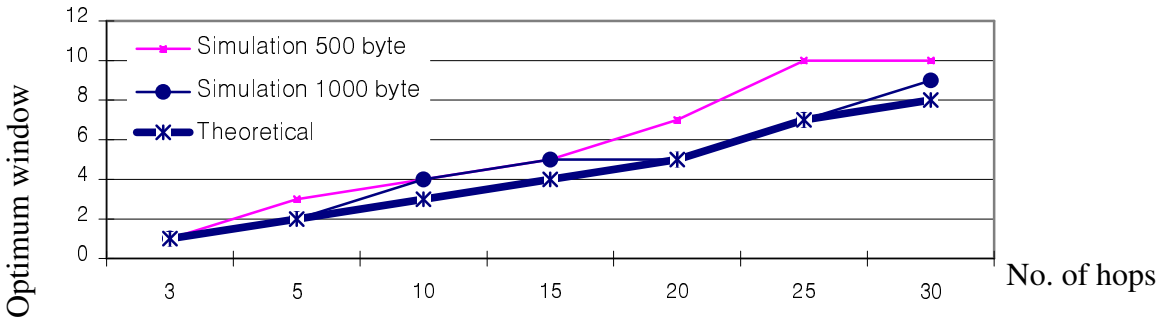


Figure 3-13. Optimum window yielding the best throughput according to the number of hops between simulation and theoretical results. Where the difference between the theoretical one (the minimum boundary values as in Equation 3-15) and the simulation ones (error ± 1) implies the presence of the offset of Equation 3-16.

3.4.7.2 Impact of packet size variation

Primarily, the optimal window varies as a function of hop length as in Figure 3-13. Thus, it verifies that the sender should consider the byte-stream advertisement as multiple units of a specific packet size. Given that consistent lower overheads[1] are required for each data packet transmission, in terms of sizing a proper *adwin*, Equation 3-15 can be rewrite with Equation 3-2 and Equation 3-3, and then the receiver can evaluate the change of the explicit window, (i.e., in view of transport layer, the receiver is only able to recognize the change of the packet size). For example, in response to probing packets whose size is different with that of normal data packets, the receiver can faciliatate the following equation in order to size an adequate *adwin* instantly. In this sense, the packet size difference in validate sampled timestamps requires further study.

$$W_{optimum,P\to P'} \geq \frac{S_{bottleneck,P'}}{P'} \times \frac{\mathrm{mod}\, ulo4 \times FLD_{P'}}{n}$$

$$\geq \frac{S_{bottleneck,P} \times B_{raw}}{P \times B_{raw} + S_{bottleneck,P} \times (P'-P)} \times \mathrm{mod}\, ulo4 \times \left(\frac{FLD_P}{n} + \frac{(P'-P)}{B_{raw}} \right)$$

Equation 3-17

Where, the packet size changed P to P', given that FLD is a minimum FLD.

3.4.7.3 Explicit window in practice

Chapter 3.4.7.1 explains how many packets can be in transit through the network *pipeline* for 802.11, supposing no collision. The *delimiter* will be thus given, namely, an *attenuation factor* ($\mathrm{mod}\, ulo4 / n$) according to the number of hops, specifically charaterized[2] by 802.11, as shown in Figure 3-14. If the *attenuation factor* is not added, then FLDmin can be set too big in cases of long hop links (e.g., > 10 hops). So, the explicit window computed may float over the sender's congestion window during the connection time (i.e., almost always congestion window-limited)—in fact, the use of the *attenuation factor* will produce slightly smaller *adwins* than the actual optimum *adwin* as evident in Figure 3-13 where the simulation results have slightly bigger *adwin* than the theoretical ones, and thus, in general, it results in slightly reduced throughput than a maximized one.

[1] If medium contention is present, the difference of the packet size affects the contention window and the NAV.

[2] If the transmission medium characteristics change, such as wireline link or a hybrid link, this *attenuation factor* should be carefully recalculated.

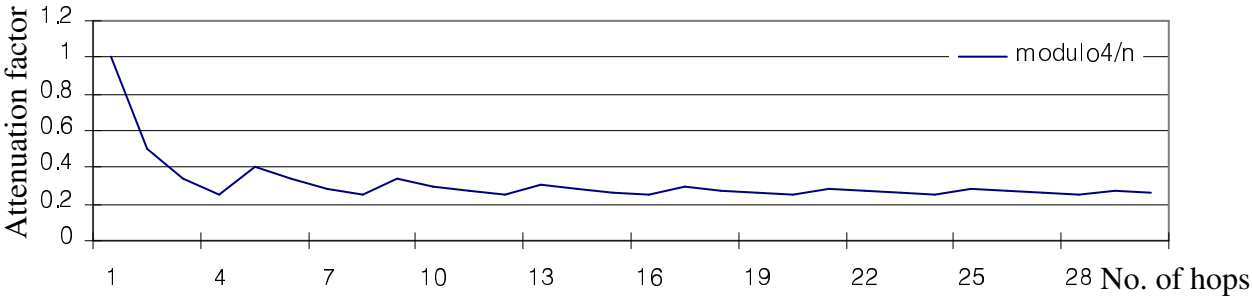


Figure 3-14. Typical attenuation factor characterized by 802.11. It fluctuates around 0.3.

The consequent explicit window to be advertised will be therefore:

$$W_{practical} = \frac{S_{bottleneck}}{P} \times FLD_{\min} \times \frac{\mathrm{mod}\, ulo4}{n} \qquad \textbf{Equation 3-18}$$

Where, $W_{LOWER} \le W_{practical} \le W_{UPPER}$ (i.e., $2 \le W_{practical} \le \mathrm{mod}\, ulo4$), and both extremes are meant to be the *reference metrics* of the least and the best for a specific ad hoc link.

The reason why variable SFLD is not used but FLDmin (thus, advertising almost always less than *modulo4*), is because we do not assure the fast convergence of the bottleneck throughput estimated and thus may get spuriously inflated or deflated bandwidth-delay product due to relatively faster varying SFLD.

And, changing packet size to the bigger, the window gets larger because of an extended FLD in spite of a reduced bottleneck throughput, but expects to be converged to an optimum sooner or later.

The term W_{LOWER} means the least window, at which the sender is under heavy medium contention, and alleviates present medium contention to an extent. Loss recovery usually relies upon timeouts if any loss occurs. Above all, the least window guarantees a minimum amount of transfer not being ceased by other heavily contending traffic loads but sustaining, less frequent but, still periodic information delivery of FLD variation from the sender to the receiver.

Both boundary values of the explicit window will be challenging metrics because the *delimiter* might advertise close to zero in occasion of heavy contention or present buffer congestion (thus,

consequently, other traffic can cease the *rater*-responding sender's transmission[1]). In terms of achieving the global fairness among identical TCPs of the *rater* in use (Chapter 5.3.4), the *delimiter* will range freely from 2 as W_{LOWER} (i.e., equivalent with the minimum *ssthresh*) to *modulo4* as W_{UPPER} to properly size the explicit windows of end correspondents in order to mitigate heavy contention and buffer related delays—beneficially, UDP traffic sources, e.g., VoIP or video-on-demand, which are congestion-insenstive, might reduce TCP flow rate because the *rater* is highly congestion-sensitive and does not make further increments unless it is allowed by a perceived FLD.

Therefore, in case, the minimum boundary value of the *delimiter* can be set intentionally to an increased one so that the sender can take the effect of the receiver's *delimiter* from a certain level of the transmission window (i.e., under that, congestion window-limited). Also, a certain level of the *buffer occupancy* (Chapter 5.1.4) can be used to discontinuously bring into play the receiver's *delimiter*. The freezer as introduced below in Chapter 3.4.8 can from time to time complement, putting the sender into the frozen state in terms of highly fluctuating FLD.

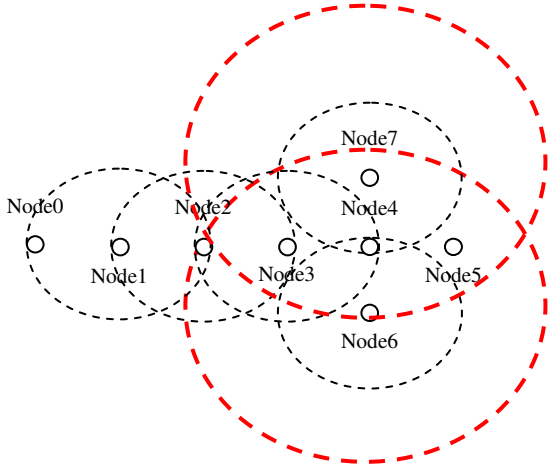
3.4.7.3.1 The rater in use

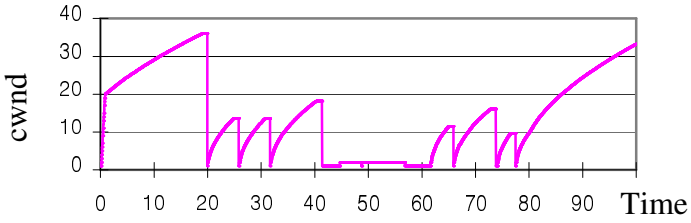
The following example as in Figure 3-15 has shown the performance of the explicit window advertisement, of Equation 3-17; furthermore, it justifies the need of the freezer. Experimented over Figure 3-15 (a), spuriously inflated FLDD and importantly the value of Figure 3-15 (e) and (f) signify intensively induced medium contention because no other degrading factors are supposed to be imposed. From Figure 3-15 (e) and (f), the sender hardly succeeded transmissions between about 42 and 63 sec because there were another interferer that almost blocked transfer of the FTP transfer. So, the sender should be frozen rather than try to transmit in penalty of packet drops, resultant exponential backoffs, and reduced *ssthresh*. In other times, the *rater* performed reasonably. When the first CBR source was only applied, it shared the leftover resources, resulting from a reduced rate appropriately.

[1] In the contrast, the normal Sack TCP tries to increment as many as possible unless a packet loss occurs because the normal reactive TCP can increment the congestion window either exponentially or linearly until it has a packet loss.

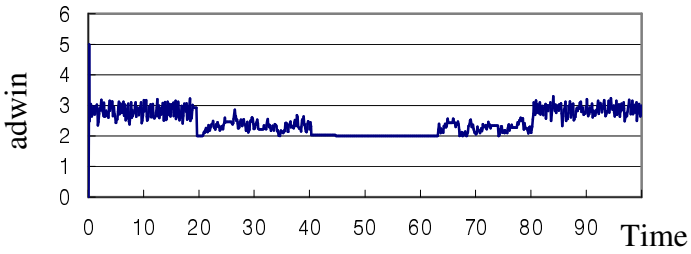
(a) A bulk FTP transfer is ongoing from node 0 to node 5 with two video sources of CBR of packet size 512B, 200Kbps. Single hop away interferences induces a certain degree of MAC contention.

One is active 20 to 80 sec from node 7 to 4 and the other 40 to 60 from 6 to 4. Both interfere with the origination and reception of node 2, 3, 4, 5 and 6 or 7.

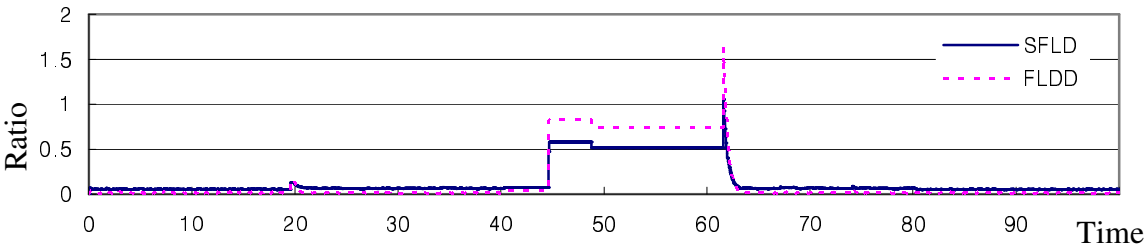




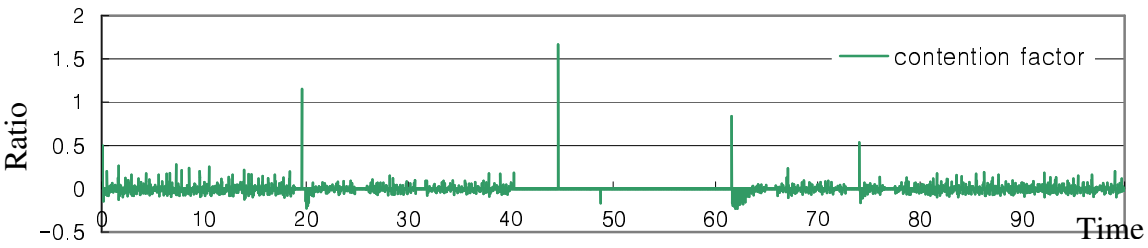
(b) Congestion window suffered by 10 timeouts.



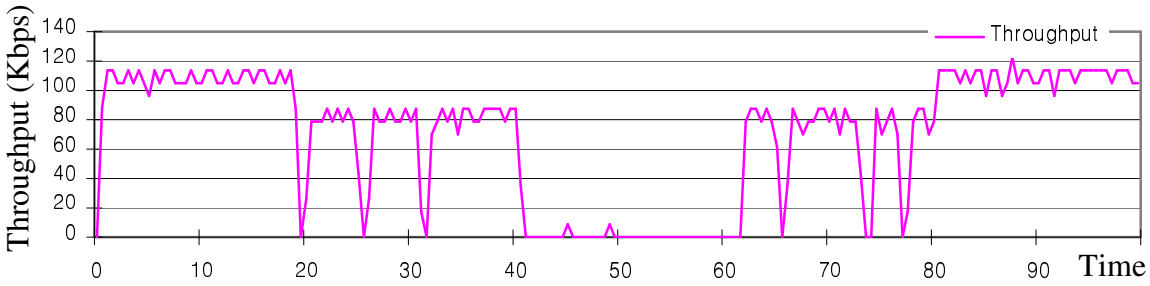
(c) Computed explicit window to be advertised from the sink. For 41.7 to 63.1 sec, it is delimited by the lower bound that is 2 for 5 hop link (meaning heavy contention period).



(d) FLDD and FLD variation likely implying packet losses



(e) The change of FLDD in portion of SFLD, $(FLDD_{i+1} - FLDD_i) / SFLD_{i+1}$



(f) Throughput measured and 6 timeouts noticeable and 4 between 41 and 62 sec of heavy contention present.

Figure 3-15. The rater in use for a FTP transfer of packet size 500 bytes with other sources. For this case, its delimiter does not have an attenuation factor in order that it forces slight high advertised windows intentionally.

As shown in Figure 3-15 (e) at a peak at 44.6 sec, when a data packet is successfully received at the receiver, it gives a sign of highly inflated FLD information compared to the previous, signifying an imminent packet loss (i.e., *contention factor*). The receiver can then inform the sender of the inflation. It is noticeably problematic when the longer hop link and the larger window available (highly fluctuating FLD), because the sender's retransmission timer set is inadequate for ad hoc links.

As a matter of fact, in case of long hop links (e.g., > (say) 10 hops), we might not assure $S_{average}$ as to be $S_{bottleneck}$ because there are non-trivial difference between $S_{average}$ and the actual $S_{bottleneck}$. So, $S_{average}$ could be too conservative to limit the sender's transmission window properly. In the meantime, the sender may suffer from premature RTO expirations.

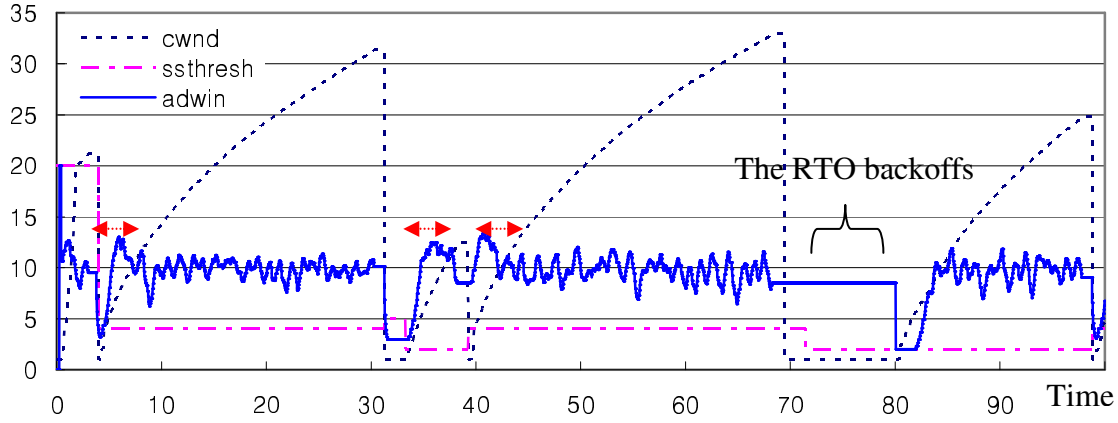


Figure 3-16. The example of the RTO exponential backoffs appeared in use of the receiver's delimiter only. (hop length 20)

As shown in Figure 3-16, not properly controlled *adwin* causes packet drops, occasionally suffering the RTO exponential backoffs as seen; falsely controlled *adwin* injects spare packets and then deflates *adwin* accordingly—Such instantaneous decrease could make an increase of the *adwin* again, and thus the *adwin* oscillates. Such flaw requires the need of the freezer, as well as the ad hoc sender enhancements (Notice that, after each loss as indicated by the arrows, the transmission window was controlled by sender's congestion window progression and the *ssthresh* was set by half the previous transmission window). The following Chapter introduces two receiver-oriented freezing mechanisms to alleviate such undesirable RTO expirations, and to avoid the resultant reduction of *ssthresh*.

3.4.8 Freezing mechanism

In terms of medium contention problem, the last resort the sender can recognize of extremely high medium contention is when the sender reluctantly timeouts. In order to proactively prevent packet losses from the medium contention-induced drop, the receiver when observing the substantial contention should propagate a freezing feedback to the effect that no further transmission by sender can reduce the contention level, expectedly avoid further packet drops, and thus save the bandwidth—the freezing options addressed here might be able to work independently of the explicit window advertisement mechanism, and coexist with the *delimiter* in order to complement.

3.4.8.1 Freezing points

- Provisional freezing – due to lessened advertised window, the sender is occasionally blocked and will not make further transmission until the sliding window permits. In the meantime, the sender cancels the retransmission timer and starts probing, immediately if no outstanding in-flight, and some time (e.g., one RTO, "rtxcur_") later if any exists. In reception of a subsequent ACK with a window update, it resumes transmission accordingly with the timer set again. The size of permitted receiver's window puts the sender into the slow start phase with the updated *adwin*, which is detailed later in Chapter 3.5.

- FLD fluctuation degree, Forward Link Delay Deviation (FLDD)—like the smoothed RTT deviation estimator (RTTVAR), we use a *smoothed mean deviation estimator,*

which is a good approximation of the standard deviation but easier to compute since no square root calculation. Its applicability for ad hoc networks is based upon the assumption that the deviation of MAC related delay is to estimate the degree of medium contention induced since the transmission and propagation delay per packet is fixed and most dominating factor is the medium contention in 802.11. The FLDD is computed as follows, like RTTVAR:

$$FLDD_{i+1} = \frac{1}{4}\left|FLD_{i+1} - SFLD_i\right| + \frac{3}{4}FLDD_i \qquad \textbf{Equation 3-19}$$

Where, $SFLD_{i+1} = \frac{7}{8}SFLD_i + \frac{1}{8}FLD_{i+1}$, $SFLD_0 = FLD_0$, and

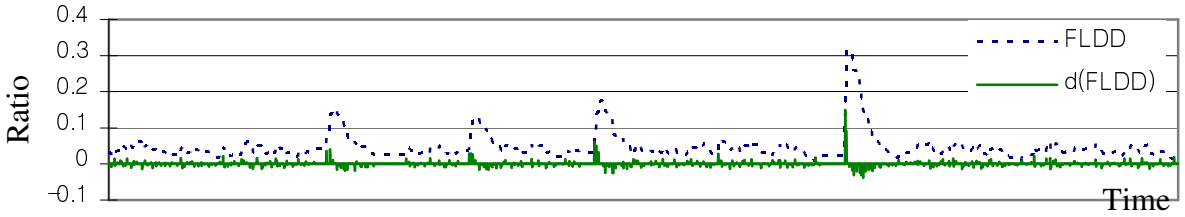
$FLDD_0 = {FLD_0}/{2}$,
$FLDD_1 = FLD_1 - SFLD_0$
, and if *n* chages, *i* goes to 0.

Notice that for simulations, we used the coefficients of FLDD and SFLD as in those of RTTVAR and SRTT, respectively (i.e., 3/4 and 7/8), but these can vary if necessary.

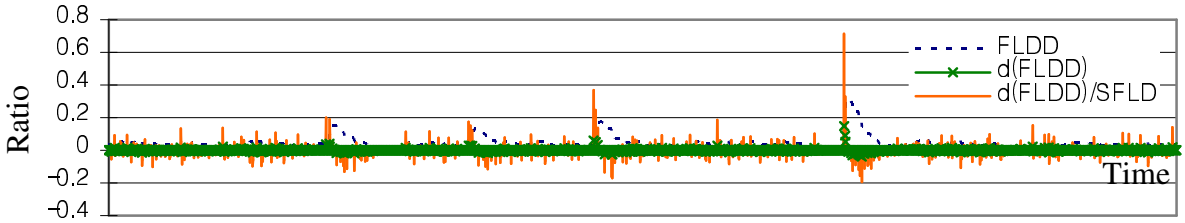
3.4.8.2 Contention factor

The presence of the *jitter effect* in the arrival times of packets, or less window expanded, due to the sub-optimal behavior of the MAC protocol does not ensure the timely arrivals of ACKs (ACKs starving). So the sender might invoke packet retransmission(s) not by fast retransmission(s) but timeout(s) in most packet drops. In turn, TCP sender may suffer from a number of invocations of the slow start phase, wasting network resources in terms of ramping up again to the available capacity (even though, as a matter of fact, it takes relatively less time because of less window expansion over the whole TCP connection time). We thus propose the *contention factor* as part of the MAC-aware congestion control. It is supposed to timely freeze the sender in order to mitigate the medium contention, whose extent is perceived by the following:

$$Contention\_factor_{i+1} = \frac{\left(FLDD_{i+1} - FLDD_i\right)}{SFLD_{i+1}} \qquad \textbf{Equation 3-20}$$



(a) The difference between consecutive FLDDs.



(b) The difference in portion of the current SFLD.

Figure 3-17. Contention factor in use to detect high FLDD variations. It implies the instances of considerable medium contention induced.

As shown in Figure 3-17 (b), the difference of fluctuating FLDD over the current SFLD, Equation 3-19 points out the prominent peaks of FLDD. The *contention factor* over a certain ratio signifies the presence of considerable medium contention. This ratio (contention factor) plays important role in warning the presence of medium contention (Chapter A.8). If the contention factor is greater than α (Namely, contention threshold, where the less α, the more frequent ZWA and vice versa), the receiver will propagate ZWA to cancel the timer and freeze until the next window update comes; in the meantime, the probing mechanim takes place in a certain time period, as addressed in Chapter 3.5.

3.4.8.2.1 Choice of α

As seen, the *contention factor* takes the difference of FLDDs, measured by two consecutively receiving packets, into account to see whether it exceeds a certain portion of current SFLD. That is, rewrting the contention factor of Equation 3-17 with alpha becomes:

$$contention\_factor_{i+1} = \frac{FLD_{i+1} - (SFLD_i + FLDD_i)}{SFLD_{i+1}} > 4 \times \alpha$$

As an example, at α 0.5, FLD_{i+1} of receiving data has to exceed about more than 3 times of $SFLD_{i+1}$ (just assuming $SFLD_i + FLDD_i > SFLD_{i+1}$) and so about 4 times of $SFLD_i$ in order to trigger freezing feedback, and at α 0.3, at least three times the $SFLD_i$. The freezing frequency is thus related to the previously sampled SFLD information coupled with the previous FLDD information so as to perceive instantaneously inflating FLD.

The hardship in choosing an adequate alpha according to dynamic hop length changes during the connection time does not allow the direct use of the *contention factor* in case of high node mobilty. Further study is thus required to find an optimum α set in sense of a *reference metric* according to a certain hop length.

3.4.8.3 Freezing timer

As a complementary approach, by the receiver's side, against the sender's sub-optimal RTO computation, the receiver will give a feedback to freeze the sender in case of premature RTO expiration predicted.

As an example, in support of a typical link-level ACK scheme employed, local retransmissions perform to reduce the number of end-to-end retransmissions. While doing so, the local retransmissions might cause unnecessary timeout(s) and, then, invoke unnecessary end-to-end retransmission(s) with drastic reduction of window (set to an initial window). In order to avoid this problem, the receiver will initiate a timer which determines when to freeze to prevent the premature timeout of the arrived packet. Afterwards, a subsequent arrival of ACK if with an updated window will unfreeze the sender and resume the transmission with simply set, or reset, timer.

3.4.8.3.1 The delivery of the freezing feedback through the reverse path link

The *freezing timer* aims for preventing the sender's spurious RTO expiration (and resultant window reduction) and mitigating MAC related delay in the meantime. As a simple principle to achieve those, if the FLD of a packet received is greater than the freezing timer initiated, the receiver sends a freezing ZWA to prevent the spurious RTO expiration of the packet.

R. Ludwig [92] addressed that, (in the sense that the *retransmission timer* is roughly offset by one RTT before triggering RTO in response to every received ACK for new data, namely

retransmission timer offset) the difference between the retransmission timer and RTO is important to determine in order to eliminate spurious timeouts because one additional RTT held is able to avoid spurious timeouts from a sudden delay of a number of link level retransmissions as long as the RTT never grows faster than the *retransmission timer* can adapt.

Hence, in terms of the receiver's freezing timer regarding the sender's RTO, the *retransmission timer offset* will be beneficially acting as to provide a spare time for timely delivery of the freezing feedback of ZWA prior to timeout occurrence.

3.4.8.3.2 Estimation of the sender's RTO

As shown in Figure 3-18 below, the *freezing timer* set might be able to comply with the change of RTO updated at the sender in the sense that the change of the forward link delay is dependent on that of the reverse link delay, because of half duplex and symmetric path in general. Thus, we believe that the receiver can timely freeze the sender with the freezing timer before an actual sender's timeout occurs. For example, when a data packet is received at the receiver (i.e., the packet sampled for setting freezing timer 4), the time difference between consecutive timestamps of the data packet and the previous packet will be inspected by the freezing timer of the previous (i.e., freezing timer 3) in order to see the likelihood of expiration of RTO 5 set.

However, due to the ambiguity of whether each receiving packet belongs to the same window of data, the difference between timestamps of consecutively receiving packets may be extremely large. Thus, taking the previous freezing timer to inspect its subsequent packet might be inadequate. In addition, the sender's historical evolvement of RTO is intuitively inadequate because of frequent link changes.

Therefore, we propose the freezing timer is to validate the FLD of each receiving packet at the receiver so that the receiver checks FLDs with a maximally allowable forward link delay timer. In other words, the receiver prepares for a timer by sampling previous FLD samples, which determines an upper bound of the next allowable FLD. For instance, a freezing timer at the receiver is initiated immediately after the sender transmits a data packet. Then, if the data arrives at the receiver, then it compares the measured FLD with the timer value in order to inspect whether the freezing timer is still pending or has been already expired (e.g., it means that in Figure 3-18, ts3 is supposed to be

equal to timestamp4). If the freezing timer is meant to have been expired before data packet arrives, it then invokes a ZWA, confirming considerable MAC related delay induced.

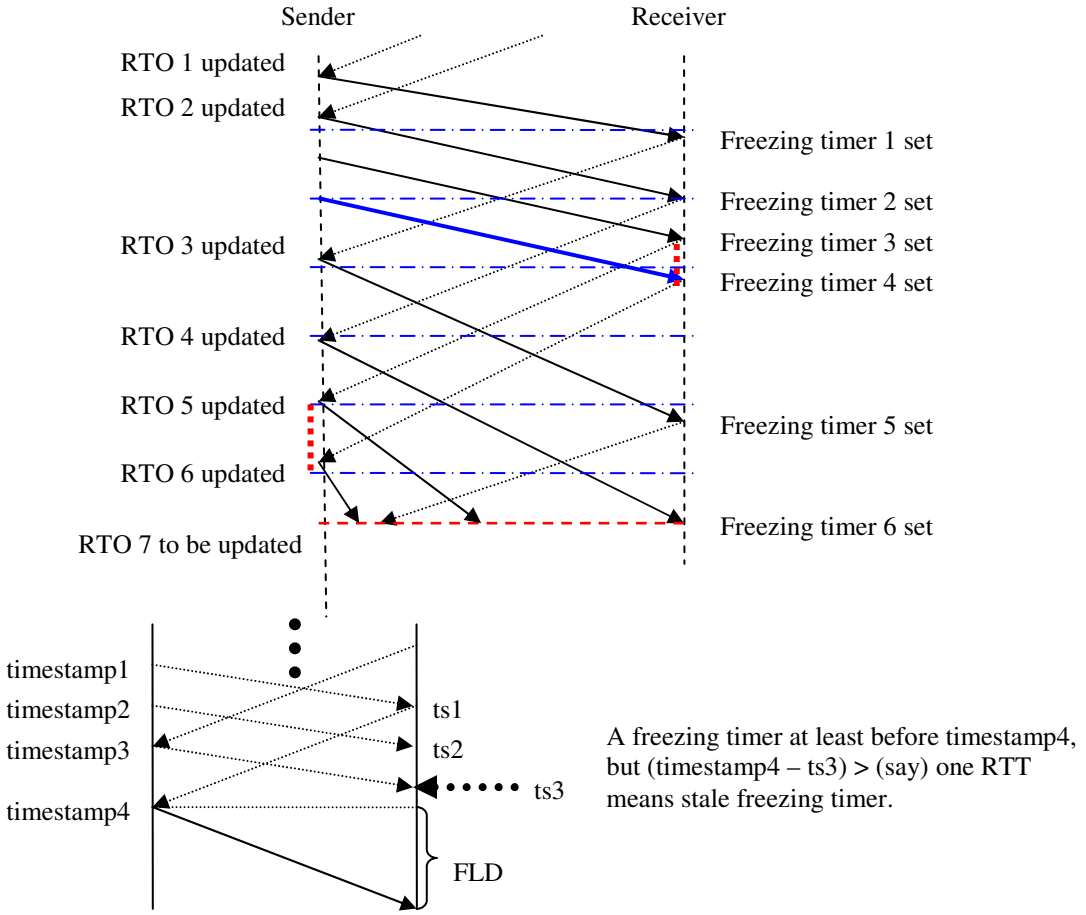


Figure 3-18. Freezing timer lagged by one FLD in order to determine a link anomaly in variation of FLDs of incoming packets.

$$Freezing\_timer(t)_i = SFLD(t - FLD_i) + \beta \times FLDD(t - FLD_i)$$

If ($FLD_i > Freezing\_timer_i$) then ZWA.

Equation 3-21

Where β can be changed in sense of a *reference metric*, we used 3, 5, and 7 for simulations.

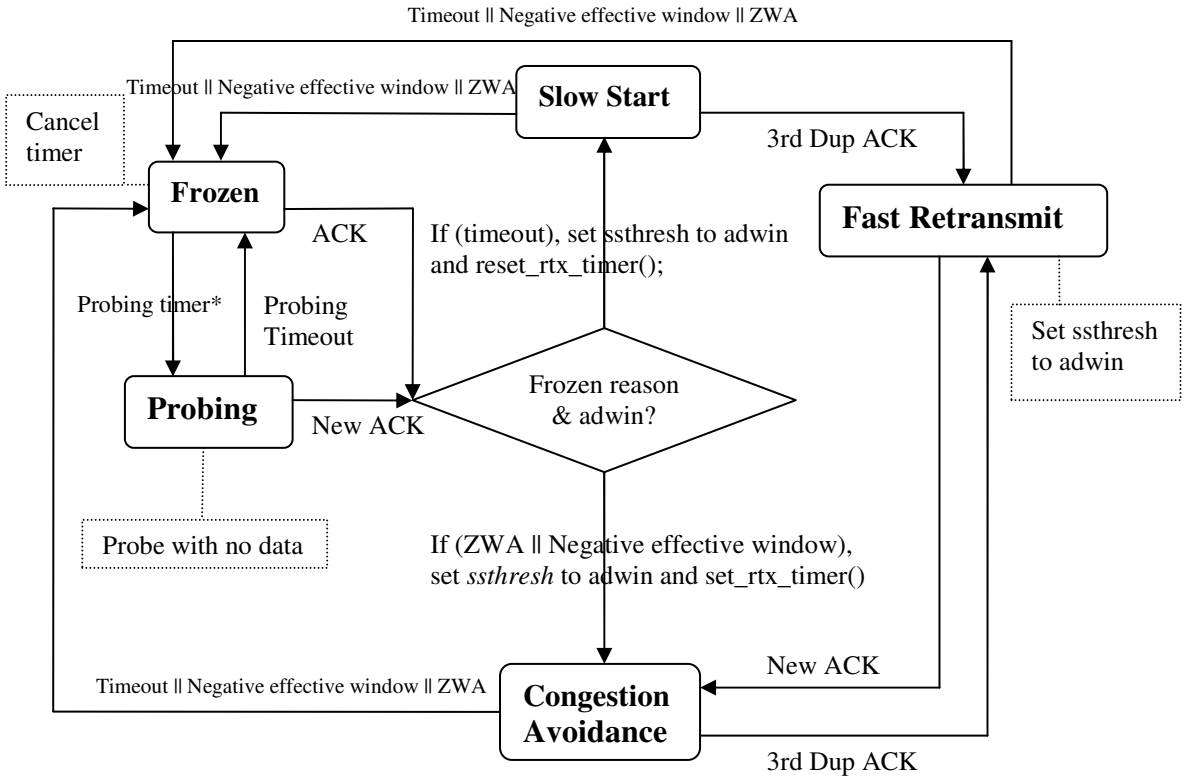
Likewise, the data packet of 'Freezing timer 4 set' is inspected by freezing timer 2, and 3 by 1, and 5 by 3. By the time setting a freezing timer each time packet arrives, time difference between timestamps of packet departure and of the time when the freezing timer set, must be no longer than

(say) one RTT (or 2*SFLD) or a certain validating interval in order to avoid a stale freezing timer in use.

By the similarity that both the contention factor and the freezing timer use historical SFLD and FLDD information, we will only evaluate the *freezing timer* and justify its use as the freezer (even though β in the freezing timer could be also problematic to be reasonably chosen in respect to network condition as noticed in Figure 5-3).

3.5 Modifications in Sender-end

In terms of mitigating medium contention-related congestion problem, the receiver has been modified to limit the receiver's window in order to prevent the sender's spurious window progression, which ultimately alleviates the number of packet drops occurred by timeouts. To an extent, it mitigates the medium contention degree so that data and ACK packets keep flowing to convey the next subsequent ACK timely to the sender. However, on an occasion, the sender might suffer from a packet absence recognized by duplicate ACKs, or bulk timeouts due to sudden route breakages or network partitioning; consequently, the sender quenches the congestion window to a small value. After that, the sender is congestion window-limited. Hence, essentially the sender should be also elaborated to get fast restored from this situation, ramping up the congestion window to an available bandwidth as soon as possible.



* Probing timer: If no outstanding in-flight packets at the frozen state, probe immediately, and otherwise probe after probing timer, and if any probing timeout experienced, frozen reason is then set to "timeout" with reset timer discarding outstanding packets.

Figure 3-19. Transition diagram for the modified ad hoc TCP sender's behavior when com-municating with the modified ad hoc receiver.

3.5.1 The type of frozen

As shown in Figure 3-19, each state is equivalent with the state of normal TCP's except the frozen state. Communicating with an ad hoc receiver, the sender is put into the frozen state in reception of three implications (i.e., an explicit ZWA from the receiver, provisionally frozen, and timeout), all of which cancel the retransmission timer due to, doubtful, imminent route breakage and give a chance to renew the *ssthresh* to be more adapted to the current path link condition.

Immediately after the frozen, a probe timer initiates to expect further subsequent in-flight packet(s) (if any) to be received. In reception of a subsequent ACK, given that the probe timer has not been expired (i.e., prior to initiating probing), the sender will be put into a proper state according to the received *adwin*. Otherwise, if it is expired and begins probing, the sender resets the retransmission timer, discarding all the outstanding in-flights[1]. Later, it goes under the slow start phase according to the *adwin* of the next successful probe echoed. In case that no in-flight packet to arrive exists, immediately it begins to probe but sustains the current frozen reason until the first probe fails.

3.5.1.1 Frozen by ZWA

In stead of a proactive detection of the link connectivity scheme in order to freeze the sender, the receiver will intentionally freeze without any explicit notification of imminent disconnection (as aforementioned, just determined by the mean deviation of FLD). After the sender realizes the route has been restored, it is put into the congestion avoidance phase with a *ssthresh* set to the advertised window size of a window update.

3.5.1.2 Frozen by TCP retransmit timeout

Due to highly moving nodes, unpredictably a link between corresponding pairs might be broken in a sudden. The modified receiver could never be aware of such sudden link breakages and also never inform the sender of the breakages because the reverse path was already broken in the sense of the common path, where the forward is as same as the reverse way. In high node mobility, the sender prefers to be frozen when a timeout occurs because timeout rather signifies a route disconnection.

[1] After the sender sets the timer again by the successful probe, it is likely to timeout unacknowledged previous window of data sent before the probe. This is why the sender should reset the timer in order to discard all the previous outstanding in-flights if ever probed with at least one proing timeout occurred.

In response, the sender, after some time or immediately, depending on whether in-flight(s) remain, starts to probe, anticipating a new route established in a certain time later. The reason that the sender waits for some time (i.e., one additional RTO-equivalent timen) to commence probing if any outstanding packet(s) remain is because the sender's retransmission timer is not adequately set for dynamic ad hoc links. If the sender waits for some more time, then it can receive the next subsequent packet(s) because the path could have been still connected. That is, the sender has to wait for additional time to probe, against premature RTO, rather than immediately slow start. Eventually, the sender can improve *goodput* as well as throughput, in spite of the expense of the additional time.

Acknowledging the probe, the sender will resume from the slow start phase with a certain *ssthresh* coarsely identified by the probe, rather than from a specific congestion window (further requiring an effort to estimate with regards to the new route rerouted and different packet size between the probe and data packets). Supposing the probe packet size is equal to the data packet, in general, the probed packet might experience a bit prolonged traversing delay because the probing is almost initiated in cases that substantial medium contention has been present. Thus it can have an inflated FLD and, resultantly, a reduced *adwin* in general.

On the other hand, after each timeout, the sender no longer has any information about the current network path link and thus should begin from the slow start phase in order to adapt to the path link.

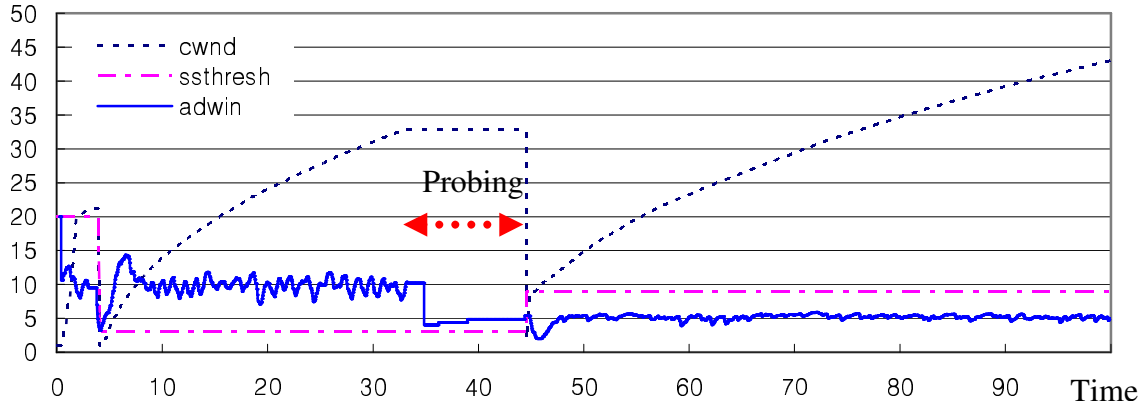


Figure 3-20. The example of the probing period superseding the RTO backoffs.

Specifically, as seen during the probing period, consecutive failures of the probe might delay the next slow start but avoid the successive failures of the slow start transmission as in (a). Once a

probe is successfully received, the *adwin* of the probed packet is taken as the next new *ssthresh*, rather than halves; notice that due to relatively high *ssthresh* set by the *adwin* received at 44.6 sec.

In brief, the use of the probing mechanism promises that the sender will not suffer from the exponential RTO backoffs because, each time it timeouts, the probing mechanism supersedes the RTO backoffs. The use of the ad hoc sender can thus reduce the number of timeouts drastically, against the premature RTOs, thus saves network resources (even though SACK option helps with faster restoration) although the frequency of the probing can be problematic.

As a result, a more flexible probing interval change will be desired to replace with, so is a challenging problem that (in stead of suffering from the exponential backoffs of successive timeouts) the probing backoff interval is now related to cope with that problem separately (being able to be adapted specifically to the ad hoc network). In this sense, when connected to other disparate networks in case, the sender may decouple the probing mechanism to enhance more adaptively.

3.5.1.2.1 Inflation of RTO on multipath routing

If the sender has in-flight packets in the network *pipeline* but the packets traverse in different paths, it might have non-sensible RTT variation. To alleviated the impact of the multipath routing, if path link, round trip links between the sender and the receiver, changes, the sender should renew the RTO and reset other terms: SRTT and RTTVAR. It can validate the values adaptable to a new path link. Moreover, the fresher timestamp of packets, the more recent path link condition can be measured. If the departure timestamp of an incoming packet is stale when compared to that of the previous packet, the sender will not take into account the packet to be sampled to update RTO. Also, the sender should not take into account the *adwin* of ACK received if stale. To investigate the sender's RTO computation to be most optimized for dynamic ad hoc network is challenging (Chapter7).

3.5.1.3 Frozen by overwhelming the pipeline

In case of a negative effective window by an intentional reduction of *adwin*, the sender will not thus transmit further packets, putting itself into the frozen state until a next packet arrives to reopen the window (Chapter 3.3.1.1). In the event of the negative effective window, if subsequent in-flight

packet(s) remain, it initiates a probe timer (gives another time period to wait for subsequent packet arrival), canceling the retransmission timer (*frozen*). If subsequent packet arrives within the probe timer, the sender's state goes to the congestion avoidance phase to start from the *adwin* updated, otherwise, it probes the receiver, setting to "timeout" as the frozen reason (all outstanding packets are supposed to be lost). In this case, the following successful probing puts the sender into the slow start phase with set *ssthresh* to the updated *adwin*. If no in-flight remains, the sender immediately probes; if the first probe is successful, it goes to the congestion avoidance phase, and otherwise, to the slow start phase.

3.5.2 Probing mechanism

By definition, TCP sender always sends a data in response to a received ACK segment and TCP receiver also sends an ACK in response to a received data segment. Suppose that the receiver propagates a ZWA in order to freeze the sender intentionally—how does the sending side know that the advertised window is no longer zero? In case of ZWA, the sender is not permitted to send any more data. If there is no in-flight packet in the network after frozen, the receiver has no way to get unfrozen with an undated non-zero window and in turn the sender has either no way to discover that the advertised window is no longer zero.

To avoid this situation, in general, the sender after reception of a ZWA should persist to send a segment (ZWP) every so often being initiated after some time (e.g., probing timer, but if no outstanding in-flights, probe immediately), and if it reports a non-zero advertised window, it can put the sender out of the frozen state. This mechanism is referred to as the probing mechanism [147], the use of which is justified in case of no packets in the network to unfreeze the sender that has been frozen already. In the meantime, all outstanding in-flight packets are discarded not to affect SRTT and RTTVAR, because it may inflate RTO spuriously. From the success of the probe, the sender should take samples to compute RTT most favorable to a route currently available.

In addition, probing interval, back-offed if probing fails, should be adaptable to the currently available route. G. Holland and N. Vaidya [74] argued about the probing interval (i.e., a fixed interval, 2, 4, 6, 15, and 30 sec) influencing TCP throughput, by which too long interval may delay the discovery of new routes and contrarily rapid injection of probes into the network may cause considerable use of network resources. Thus, they envisioned that an adaptive probing interval, for example, might be a function of RTT, which could be a more sensible choice. However, it could be

also flawed if the round-trip path link switches to a different one on the way of probing. We thus put this aspect to a further study in order to find an optimum probing interval.

In a high medium contention period, the higher backoff timer (interval), the higher RTO could be obtained if the probe successfully round-trips. It means that the probe-induced inflated RTO could be sometimes problematic because if the following packet is lost it results in a prolonged loss recovery time due to the longer RTO. Thus, the probing timer should be limited by an upper boundary. For example, the least available throughput ($S_{\min}$) can be facilitated because it is specific to the current hop length and could determine an allowable, maximum RTO; A maximum probing timer could be bounded to a maximum RTO ever observed since the hop length changed (or twice the RTO for the sake of reducing probing frequency, or far longer because of the likelihood of hop length changed to a longer one, whose RTT exceeds the maximum boundary). However, its efficiency in the sensitivity of the throughput perspective is put into a further study because more frequent probings require the substantial use of network resources and, for instance, will be unnecessary in case of long-term network partitioning.

As in our study, if it is not successful for a probe to return within a certain timer set, the probe timer back-offs exponentially up to 32 seconds at most. If set the maximum backoff interval to a fixed small one, e.g., 2 seconds [104], for simulations of stationary lengthy chains (greater than, say, 20 hops), the probe hardly round-trip such lengthy routes within 2 seconds. For dynamic topologies where the longest hop length is 10, it might be set to a smaller one; therefore, the sender should be aware of hop length to the receiver in order to confine a maximum probing backoff interval more properly.

Chapter 4
Experimental Methodology

4.1 Methodology based on the network simulator

4.1.1 Introduction to the Network Simulator

All simulations experimented for the study were performed by a popular, pervasive discrete event simulator, network simulator (version ns-2.1b9a) [106], available from Lawrence Berkeley national Laboratory (LBNL) with the extensions of the MONARCH project at Carnegie Mellon, the extensions of which include a model of mobile node for ad hoc network as in Figure 2-1, a set of routing protocols, and mechanisms to model node mobility (e.g., use of "setdest") to be fed to the simulation at run time, more detailed in [37, 64, 71].

OTcl [111] provides a control and configuration interface to the users for simulation runs and easily by binding variables (e.g., ns extension by example is found in [108]), C++ member variables (ns C++ class hierarchy found in [107]) identical to OTcl instance variables can be accessible, then users can set and monitor the variables using simple tcl scripts [114].

In addition, the *ns* has visualization tools such as network animator (nam) and graph generator (xgraph) to help the users analyze their simulation results.

4.1.2 Modifications made to the ns-2 simulator

In terms of the scheme proposed in this thesis, the following changes were made onto the baseline TCP "sack1" (i.e., "tcp.h" and "tcp-sack1.cc" in the ns-2 simulator and the addition of "adhocsink.{cc,h}" of two classes, "Sack1AdHocSinkClass" extended from TclClass and "AdHocSink" extended from TcpSink). The simulation scripts and full C++ codes are available in the CD-ROM attached at the back.

4.1.2.1 Basic modifications of the normal baseline TCP sender "sack1.cc"

Because of absence of a responding mechanism against incoming ZWAs originated from the ad hoc receiver, the followings are implemented as added to a normal baseline TCP.

Timer will be cancelled in reception of ZWA from the receiver. Afterwards, typical probing begins after some time (i.e., one intial probe timer of the current RTO), and if lost, it backoffs exponentially up to 8 secs at maximum. After a window update comes in, it simply sets the timer (i.e., set_rtx_timer())—such a simple set of the timer can likely suffer from timeout of any subsequent in-flight(s), thus it will have a conditional term to determine whether to discard, as detailed in Figure 4-2 below of the ad hoc sender's modifications.

4.1.2.2 Ad hoc Receiver TCP modifications

The "adhocsink.cc" and "adhocsink.h" were added in ns2 to form an Ad Hoc sink agent, rather than the current "sack1sink.cc" was modified. The details are illustrated in Figure 4-1.

- The function, "recv()" at sink monitors the timestamp of incoming data packet in order to compute the forward link delay. Given that the number of hops is informed from the lower layer (IP header), it calculates several values, FLD, FLDV, SFLD, and FLDD, and then calls a function, "adwindow()" to compute the explicit window, contention factor, and freezing timer. Afterwards, it designates the computed advertised window in the TCP header of outgoing ACK (the term, "adv_win_" added to "tcp.h" to be used at the sender TCP).

- The sampling of the timestamp at sink is irrespective of sequence number. Recently arrived packet whose timestamp information is fresher than previously received one is only taken into account, and so if one comes in stale, the sink does not sample its timestamp because recently traversing packet gives fresher timestamp information and can determines more recent link condition.

- When a probe that has a specific sequence number (-1) is received, the function, ack() updates the advertised window if available and propagates with the saved sequence number last successfully ACKed. For faster round trip time and saving bandwidth, we use a probe

without TCP data payload and so it gives relatively a small FLD along the path link. The receiver could use Equation 3-17 to properly dimension the *adwin* according to the actual data packet size, but due to the likelihood of having a new path, the receiver no longer uses the current historical SFLD information in responding for the next *adwin*. For this study, its FLD information will not be sampled for SFLD and FLDD, and so the *adwin* in response to the probe will be set to a *modulo4* unless otherwise stated, as in Equation 3-16, and acts as a new *ssthresh*. The inadequacy of such an *adwin* can be retrieved by some other times (e.g., ZWA, provisional frozen, and timeout) to renew the *ssthresh*.

- Receiver–oriented probing after propagating ZWA might be necessary if no reliable probing mechanism is available at the sender even after frozen. If the sender was frozen by the receiver in case and, unfortunately, there is no more in-flight packet to unfreeze, it should have to unfreeze the sender by probing at the receiver with a replica of the previously sent ACK and, if available, with an updated advertised window. For this study, the sender-side probing mechanim was implemented.

- The freezing timer requires 2-dimensional array where the first column stores the arrival time of data packets received. Then it is put together with the second column that contains the value of the freezing timer computed each time data packet is received. In the meantime, the FLD of subsequently receiving data packet is inspected by a freezing timer whose first column value is equal (or closest) to the timestamp of departure time of the received packet.

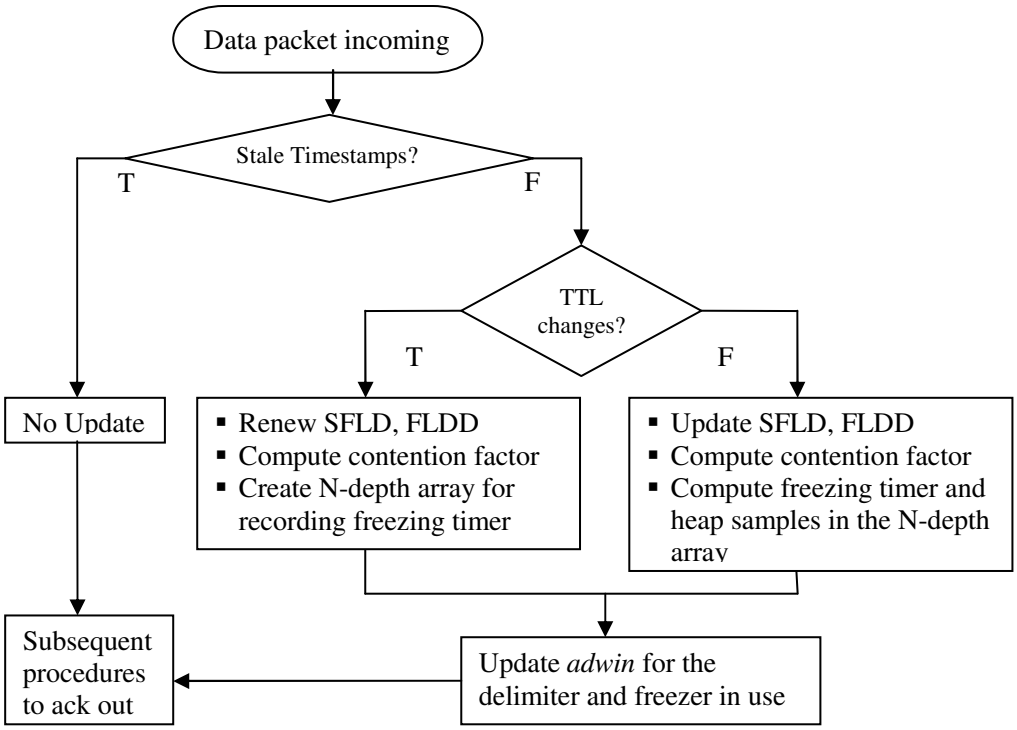


Figure 4-1. Summary of the ad hoc receiver's ns implementation. When a probe packet, whose size is smaller than that of the actual transmtting data packet, is received, the measured FLD information will not be taken into account.

4.1.2.3 Ad hoc Sender TCP modifications

The "tcp-sack1.cc" was modified to have a probing capability and response with changing advertised window received (as well as ZWA both freezing the sender and resetting the timer of outstanding in-flights). The detailed flows are illustrated in Figure 4-2.

- The function, "recv()" at sender reads received advertised windows (i.e., the term, "adv_window()" in "tcp.h") and then delimits the next transmission window.

- On an occasion, if a zero window acknowledgement is received, the sender resets the retransmit timer. Then, it sets a probing timer so that after it timeouts, the sender sends a probe to check the link connectivity.

- The sender implemented the standalone probing mechanism whose timer is set each time ZWA is received or timeout occurs. In the cases, if no in-flight packet in the *pipeline*, it

probes immediately (i.e., probing timer is zero) and otherwise waits for a probing timer of the current RTO.

- For the following cases, ZWA and provisional fozen as well as fast retransmit and timeout, the sender updates its *ssthresh* with the next incoming advertised window that convinces the current link sustainability. So, the sender from time to time calibrates its transmission rate.

- Received ACK whose timestamp information is fresher than previously received one will be only taken into account to update the receiver's advertised window, and so if a stale one comes, the function, recv() does not take its timestamp into account.

- If the sender communicates with a wireline receiver, it will perform as the same but does not take the advertised window into account.

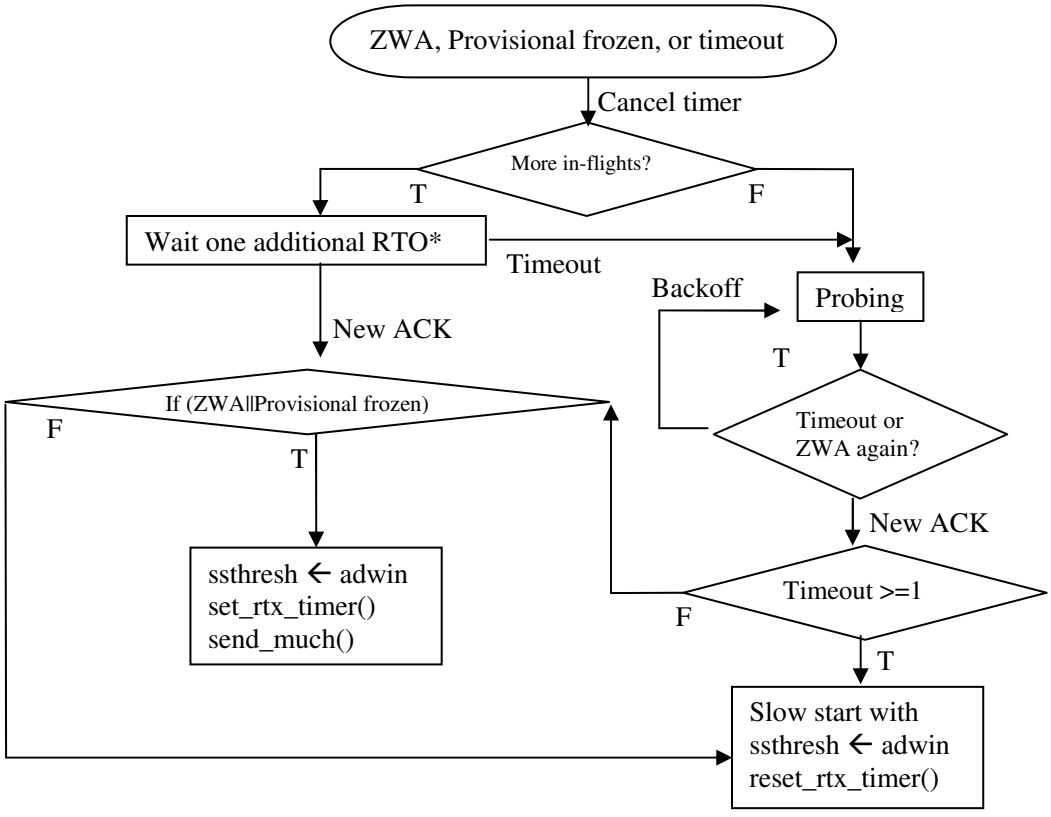


Figure 4-2. Implementation summary of the ad hoc sender on receiving a TCP ACK in simulator implementation. *One additional RTO is in use in order to prevent premature RTO because the sender's RTO was not optimized for dynamic ad hoc links.

Chapter 5
PERFORMANCE EVALUATION

5.1 Performance metrics

5.1.1 Throughput

This metric is simply defined as the total number of bytes received at the receiver divided by the total time the sender was active.

5.1.2 Goodput

This metric, *goodput* is defined as the ratio of total received bytes to total sent bytes (i.e., packet delivery ratio), implying the amount of retransmitted data from the sender. It is important to determine the protocol performance in terms of battery-constraint mobile nodes and so protocol efficiency because, the less goodput, the more bandwidth wastes.

5.1.3 The number of invocations of the slow start phase

Since one of the most important objectives in a wireless scenario is to avoid unnecessary timeouts, the number of unnecessary timeouts represents efficiency in use of available network resources. Followed by our sender's modifications, two other conditions may put the sender into the slow start phase, such as either ZWA or, provisional frozen case after a probe timeout. However, timeouts only caused by premature RTO expiration will be put into account because the latter two cases are independent of the RTO-related timeouts.

5.1.4 Deterministic Buffer occupancy

The receiver can assess the *buffer occupancy*, as seen in Equation 5-1, along the path link instantaneously by means of subtracting the current advertised window from the best window (i.e.,

for this study, a *modulo4* for the current hop length). Within ad hoc networks, this *buffer occupancy* is supposed to estimate the degree of medium contention (exemplified in Appendix A.5), as well as that of the actual buffer queuing (i.e., In general, the baseline results in high network buffer utilization).

$$Buffer\_occupancy = (Expected\_throughput - Actual\_throughput) \times FLD_{\min}$$
$$= \left(\frac{n}{FLD_{\min}} - \frac{n}{FLD_i} \right) \times FLD_{\min} \times \frac{\mathrm{mod}\, ulo4}{n} = (n - W_{actual}) \times \frac{\mathrm{mod}\, ulo4}{n}$$

Equation 5-1

Where, n is hop length, FLDi is updated every receiving data packet, and FLDmin is renewed each time n changes, and $\mathrm{mod}\, ulo4 / n$ is an *attenuation factor* according to the hop length.

The *buffer occupancy* is useful and necessary once the receiver connects with a wireline sender (further addressed in Appendix A.4) in terms of deciding the time to invoke our the *delimiter* for the purpose of the buffer congestion avoidance. By similarity with the transmission window controller of TCP Vegas [3, 4], if the perceived *buffer occupancy* exceeds a threshold predefined, it invokes the *delimiter* to work in order to lower the congested buffer level, and then disables it when the *buffer occupancy* goes below another threshold, which is similar to the two thresholds of TCP Vegas either to increment or decrement. That is, the reason why we use this term when interconnecting with a wireline sender end is because the path link along the wireline link has built-in queuing level whose presence is ordinarily tolerable. Whereas, unicast within an ad hoc network does not allow the buffer queuing seperatly, as aforesaid, because of the use of the comprehensive MAC related delay.

Practically, the moving average for the *rater* was used to reduce jittering.

$$Buffer\_occupancy = \left(\frac{n}{FLD_{\min}} - \frac{n}{SFLD_i} \right) \times FLD_{\min} \times \frac{\mathrm{mod}\, ulo4}{n}$$

Equation 5-2

Where, n is the current hop length, FLDmin is renewed each time n changes, and SFLDi is updated every fresh timestamp of receiving data packet.

The least and the most *buffer occupancy* per a specific hop length will be *reference metrics* in order to dimension two thresholds appropriately to identfy path link anomalies, unlike that TCP Vegas has two defined fixed thresholds during the connection time and so works regardless of the path length changes. Further research might be required to find out their practical values for deployability, in order to exploit the *delimiter* in the wireline networks.

5.1.5 Statistical Analysis of the simulation results

The perfromacne metrics (i.e., Goodput, Throughput, the numebr of unnecesary timeouts, and other significant values presented through specific simulations) required the use of statisticial analysis for meaningful interpretation of the results as well as comparson among the performances of the schemes. Unless otherwise stated, when comparing results, the statistical values, each of which has a small percentage difference, validate reliable comparisons, given the variability observed.

We executed five simulation runs with a different seed applied each time, and then the mean value of each metric is calculated. After that, the relative error is calculated and if it is above 5% a new simulation run is executed. This process is repreated until the error is less than or equal to 5%.

5.2 Performance model

5.2.1 The Baseline

The "sack1" TCP in ns2 is used as a baseline TCP with the following features shown in Table 5-1. Unless otherwise stated, nodes use the AODV routing protocol to find and maintain the route between end nodes, which thus imposes a certain amount of routing expense in every instant of forwarding, and the 802.11 MAC protocol is used. Other simulation configurations are set as in Appendix B.1.

The baseline TCP that is fully *congestion window-limited*, will be compared with the proposed TCP scheme that properly throttles the sender's congestion window in terms of bandwidth constraint ad hoc links. The performance gain will be presented and analyzed in order to verify the inefficiency of the sender-only control of the congestion window. The resultant drawbacks of the baseline will clarify the merit of our TCP scheme.

Features	Availability
SACK	ON
Initial window	1
Fast retransmit and recovery	ON
MSS (and TCP/IP headers)	1000 bytes (and 40 bytes)
Timestamps	ON
Delayed ACKs	OFF
Advertised window	32
Maximum *cwnd*	60
Initial ssthreshold	20
Timer granularity	10 ms

Table 5-1. Features in the baseline TCP. Finer timer granularity is preferred to express the sensitivity of change of FLD, and other contingent measurements.

5.2.2 Mobility applied

Given a movement pattern, the number of changes of hop length during the connection time will be differed by different TCPs applied. This implies that according to a kind of employed TCP, the connection imposes different amount of routing overheads, which is evaluated in Chapter 5.3.2. TCP performance is determined by throughput and *goodput*, measured for the total active time, which are deterministic metrics to assess the inefficiency of an employed TCP.

With several combinations of our schemes in comparison with the baseline, a set of simulations has been performed to give analyses in view of throughput and *goodput* effectiveness when a single pair of source and destination was present as well as multiple pairs. The node movement pattern was characterized by the random way point model in variation of node max. speed, and according to the node speed, totally different node movement patterns are generated.

5.2.3 Contention applied

To evaluate the performance of rate converging against the presence of background competing traffic, we will add congestion-insensitive flows (i.e., UDP CBR sources, e.g., 200 Kbps, packet size 512 bytes). Each UDP flow when applied is of a single hop away transfer to cause the levels of consistent medium contention and to hamper the reception or origination of one or two-hop distant nodes. It aims to cause link failures, sporadically or frequently.

5.2.4 Effectiveness in throughput

Bulk data transfer using a FTP source represents a continuous flow of data packets of the maximum size allowed by the network, and will be used to determine whether the employed rate controller(s) perform reasonably well in terms of the throughput and the global fairness among competing FTP sources.

5.3 Performance Evaluation and Analysis

5.3.1 Stationary topology

The following experiements in this section were performed over stationary string chain in variation of hop length. Each node is identically distant (200m) from one another and cannot interfere with two-hop away adjacents. The packet size is 1000 bytes and each node has enough buffer size (i.e., 50) not to experience buffer overflow. Simulation time lasts 100 seconds unless otherwise stated.

5.3.1.1 The use of the congestion window delimiter

Table 5-2 shows the performance of the *congestion window delimiter* in comparion with the baseline. The *delimiter* performed to yield better throughputs, up to 47.3 %, and fewer timeouts over all the different hop lengths. Whereas, the baseline overwhelmed path link capacity and so suffered from a number of fast retransmits and timeouts, because the baseline increments its transmission window each time when an ACK is received until it experiences a packet loss. So, even though no BER and mobility is applied, intermediate routers suffer from high medium contention and as a result drop a number of packets.

Hop length		**5**	**10**	**15**	**20**	**25**	**30**
Delimiter	Throughput (Kbps)	128.5	77.9	77.8	63.7	54.9	52.6
	Fast Retran	0	0	0	0.1	0	0.4
	Timeout	2.2	4	2.5	4.9	7.4	7.1
Baseline	Throughput	111.4	59.5	52.8	54.3	51.9	43.7
	Fast Retran	37.9	8	6.1	5.1	4.4	3.5
	Timeout	6.1	15.3	13.7	12.2	11.9	10.8
Throughput gain (%)		*15.4*	*30.9*	*47.3*	*17.3*	*5.8*	*20.4*

Table 5-2. Throughput gain according to the hop length. Packet size 1000 bytes for 100 seconds simulation time.

In case, extremely induced medium contention may block packet forwarding and so causes TCP to timeout unnecessarily. The ELFN mechanism can be employed and occasional ELFN signaling is invoked to avoid the premature RTO expirations by putting the sender into the persist state in the case of the medium contention. Then, the sender begins probing. That is, ELFN could be beneficial in alleviating successive timeouts even though mobility is not applied but only medium contention, because the inference of broken link relies upon the behavior of MAC protocol.

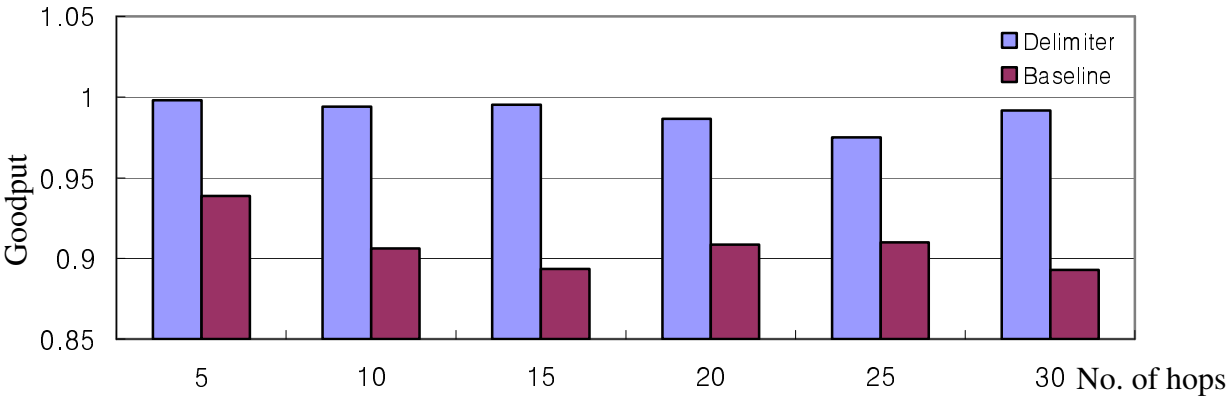


Figure 5-1. Goodputs of the delimiter and the Baseline (up to 11.4 % improvement).

As expected, Figure 5-1 appears 11.4 % goodput gain against the baseline. This is because the *delimiter* throttled the sender' transmission window according to the degree of contention, which is perceived by the FLD information of receiving packets. At hop length 30, it had a better *goodput* compared to that of hop length 25. This is because, as in Figure 3-13, there is a marginal difference between the threoretical and simulation ones of the optimum window, to which the *delimiter* can evolve up. That is, the *delimiter* at hop lenth 30 throttled the sender with relatively smaller *adwins* than the actual, and thus yielded slightly higher *goodput*.

For the *delimiter*, in general all the packet losses are perceived by timeouts because of not enough packets in the *pipeline,* as well as contention-induced postponement of packet forwarding. Therefore, it justifies the use of freezing mechanism so that if a packet experiences that path link condition changes, the receiver propagates a ZWA to freeze the sender with timer cancelled until the next packet comes in to release, and thus to avoid the premature RTO expiration to a certain degree.

5.3.1.1.1 Comparision with the optimum window

Table 5-3 and Figure 5-2 show that in different hop lengths applied, the throughput obtained by *delimiter* looks similar to that of an optimum window according to a specific hop length.

Hop length	5	10	15	20	25	30
Optimum window (simulation)	129.8 Kbps (2)	92.4 (4)	81.8 (5)	77.2 (5)	69.2 (7)	54.9 (9)
Optimum (theoretical)	129.8 (2)	85.9 (3)	80.8 (4)	77.2 (5)	69.2 (7)	53.5 (8)
Delimiter	128.5	77.9	77.8	63.7	54.9	52.6

Table 5-3. Throughput measurement for optimum windows (theoretical and simulation) and delimiter, and the size of the optimum window in the round bracket (error ± 1), over stationary string topology with different hop lengths. Each node 200 m apart, with no other contention source. Packet size 1000 bytes and 100 seconds simulation time.

Particularly for higher hop lengths, it was slightly decreased because higher hop length causes highly fluctuating FLD (and SFLD) and produces unstable bottleneck throughput. In turn, such uncontrolled packet transmisisons caused a number of timeouts. As a result, the sender quenched *ssthresh* more frequently and so slowed down the *cwnd* progression, resulting in late ramping up to the available bandwidth.

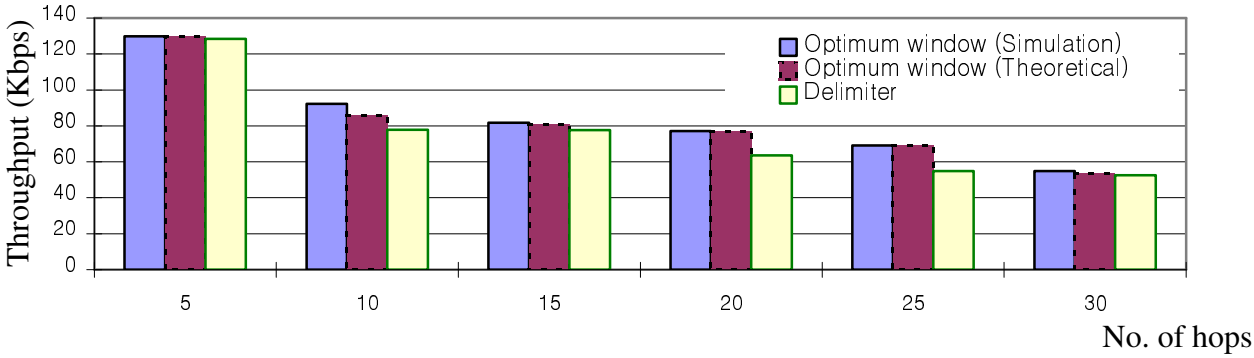


Figure 5-2. Throughput measured over stationary string topology in variation of hop length of each node 200 m apart, with no other contention source. Equivalent with the result of Table 5-3.

If the *delimiter* works with the ad hoc sender enhancements, the ad hoc sender takes into account the *adwin* as the estimation of the available bandwidth and, with that, ramps up more quickly. In addition, the following freezing mechanism alleviates the impact of the sub-optimal behavior of the *delimiter* to a non-trivial degree.

5.3.1.1.2 A combination of the delimiter and the freezer

The *freezing timer* palys role in alleviating the RTO backoffs, and unnecessary *ssthresh* reduction. For lengthy chains of single TCP connections, its outcomes, in Figure 5-3, did not demonstrate prominent throughput gain. It means the freezer invoked a number of ineffective freezings (i.e., frequent intermittent periods by frequent ZWAs (e.g., β at 3), which cancel the retransmission timer each time, may degrade throughput to some extent that they postpone the next packet transmission, and thus result in degrading *goodput* relatively as seen at hop length 30 in Figure 5-4). In case, some of in-flights could be timeouted some time later after the retransmission timer was set again as soon as a probe was acknowledged. In the meantime, the sender will experience degraded throughput.

In particular, frequent probings, initiated on the way of the freezing mechanim, may cause the sender to sample inflated RTT[1]s (i.e., frequent successes of prolonged probing time) and thus to yield inflated RTO. In turn, the sender is more likely to suffer from the inflated RTOs when back-offed. Therefore, a reasonable freezing frequency is a significant aspect in validationg the beneficial use of the freezer in terms of timely fair freezing and, importantly, periodic up-to-date *ssthresh* after each freezing. The change of β for the *freezing timer* can adjust the frequency of freezings and should be according to the level of present medium contention and the hop length of path link.

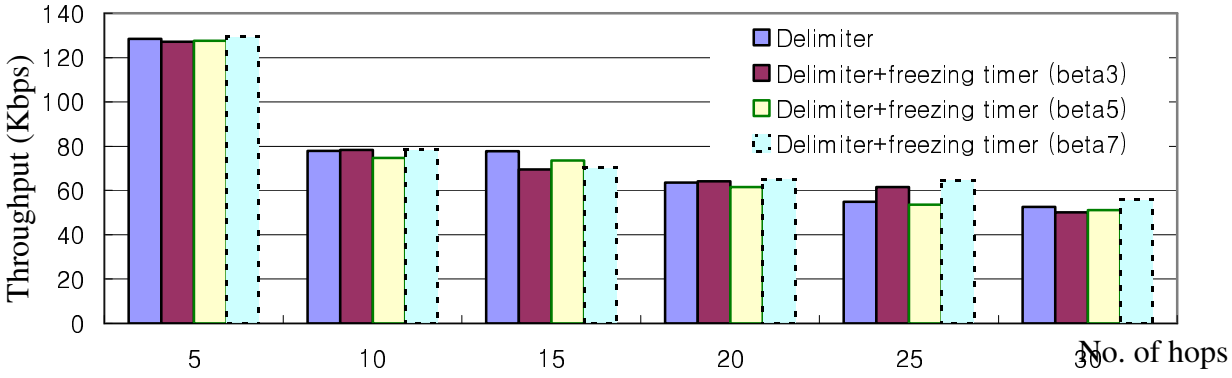


Figure 5-3. Throughput with the use of the freezer with different βs in multihop stationary string path with no other contending traffic load applied.

[1] For this study, we use probes without TCP data payload, and so the probe can round-trip faster than TCP data packets. By the way, ATP [139] disables the RTO and recovers all packet losses by the extended SACK option, and fast retransmits.

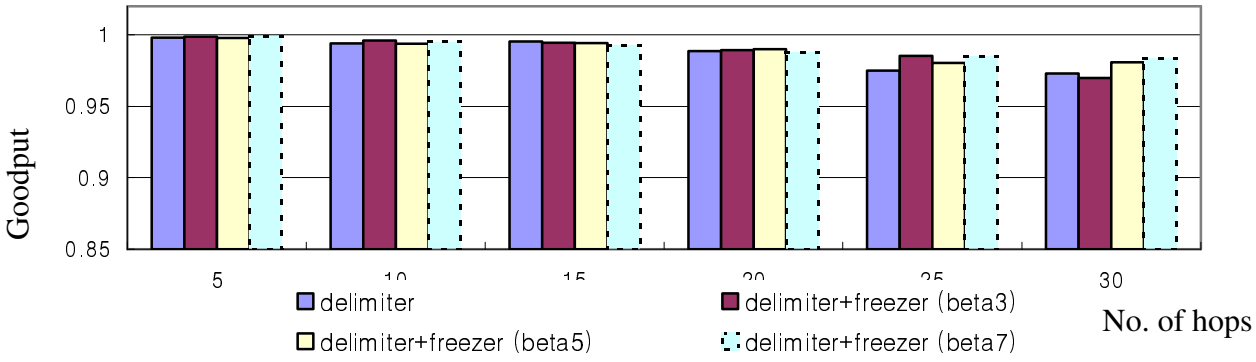


Figure 5-4. Goodput with the use of the freezer with different β s in multihop stationary straing path with no other contending traffic load applied.

5.3.1.2 The ad hoc sender enhancements

The following Figure 5-5 evaluates the performance gain by combinations of our enhancements in variation of the number of hops. All the compositions having the ad hoc sender enhancements do not seem to perform remarkably in comparison with the *delimiter* only, but does. As shown in Table 7-1, employing the ad hoc sender enhancements yielded relatively fewer timeouts and obtained roughly better *goodput*.However, particularly, a combination, the *delimiter* with the ad hoc sender enhancements and the freezer (then, we refer to as the *ad hoc TCP*) has performed promisingly in terms of *goodput*, as seen in Figure 5-6. Employing the freezer, in a variation of β, improved *goodput* relatively for longer hops.

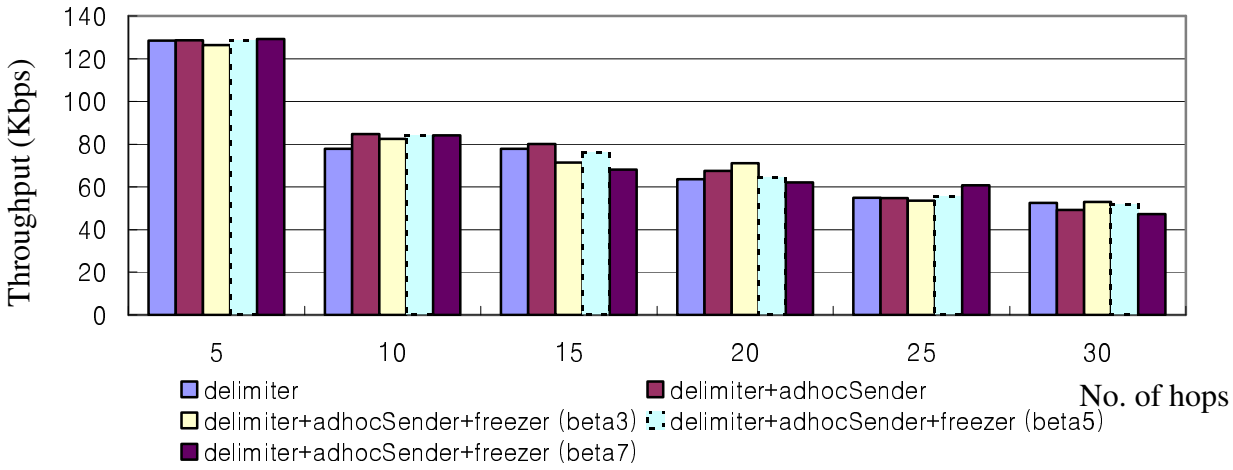


Figure 5-5. Measured TCP throughputs. The use of modified ad hoc sender with the receiver's enhancements in no other traffic loads. Notice that the use of the sender mitigates the number of timeouts. Experimented in stationary string paths, packet size 1000 bytes, 100 sec simulation time.

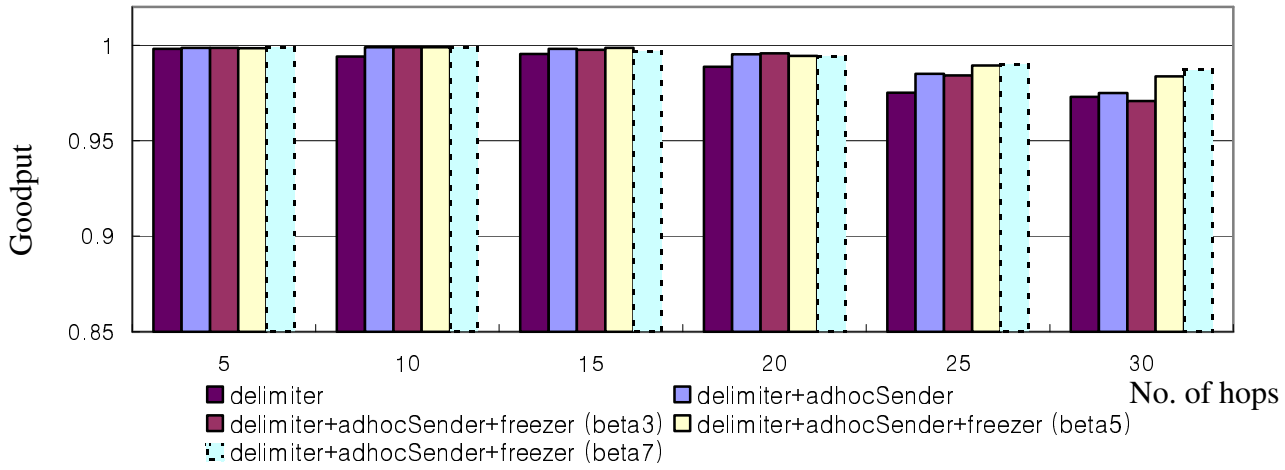


Figure 5-6. Goodputs of combinations of proposed schemes of Figure 5-5.

We will use 5 for a moderate β, in terms of freezing and resultant probing frequencies, for the simulations of dynamic topologies; however, it could be changed. Figure 5-7 compared the ad hoc TCP with 5 for β with the baseline. The best *goodput* achieved by the *ad hoc TCP*, floating close to 1, is desirable for use in power-limited ad hoc networks because of the least packet drops, so it is effective to alliveate unsuccessful medium access of overinjected packets.

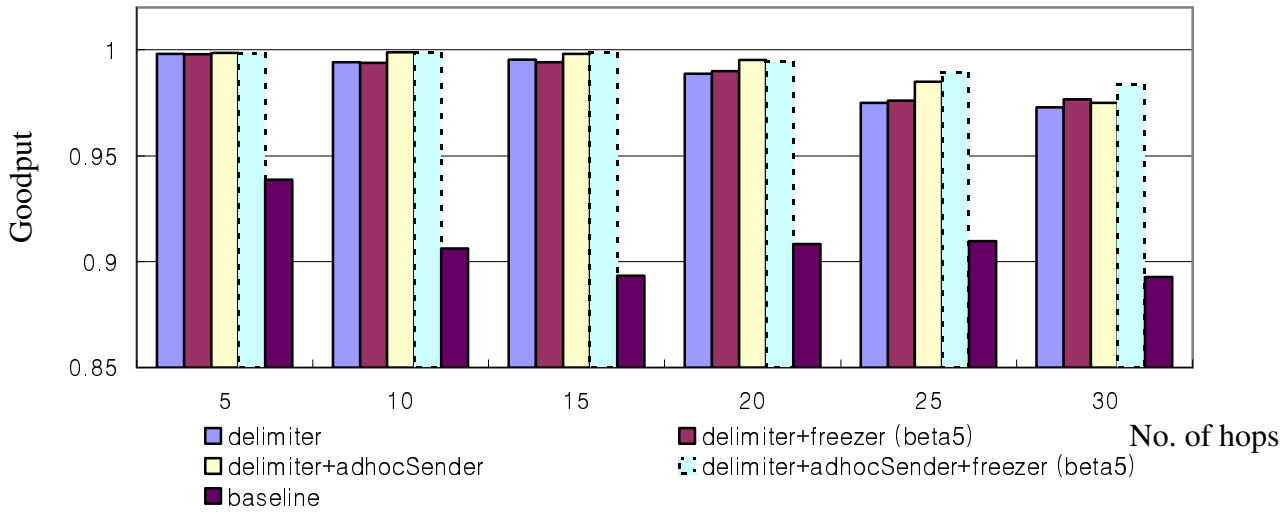


Figure 5-7. Measured goodputs of a set of combinations. Any of propositions employed improves TCP goodput. Most promsingly, delimiter, ad hoc sender, and freezer applied perform best in the goodput perspective provided that as in Figure 5-5, throughputs are almost same each other.

Whereas, the baseline took additional injections by increments of *cwnd*, probing further leftover network bandwidth until a loss occurs, so it is not effective. In the meantime, other contending source may suffer from such injudicious medium access, further inducing contention and,

occasionally, flooding network-wide routing control packets. Again it could degrade aggregate throughput, as well as *goodput*, hampering other simulataneous transmitters not to occupy the medium.

On the contrary, the proposed *ad hoc TCP* has relateviely high packet delivery ratio because the sender avoids transmitting further packets unless it perceives available bandwidth. The rationale behind this is that the *delimiter* of the receiver-end responses to a delay absorbed by a bottleneck link and computes the available utmost window in an identical manner (In this sense, the global fairness among competing sources tends to be achieved [139]). That is, every node in the network works fairly and does not try to steal bandwidth by means of the injudicious injection as the baseline does.

The following Chapter evaluates such typical situation to clarify the adverse impact of the baseline window progression when applied to moving topology, as usual cases in ad hoc networks.

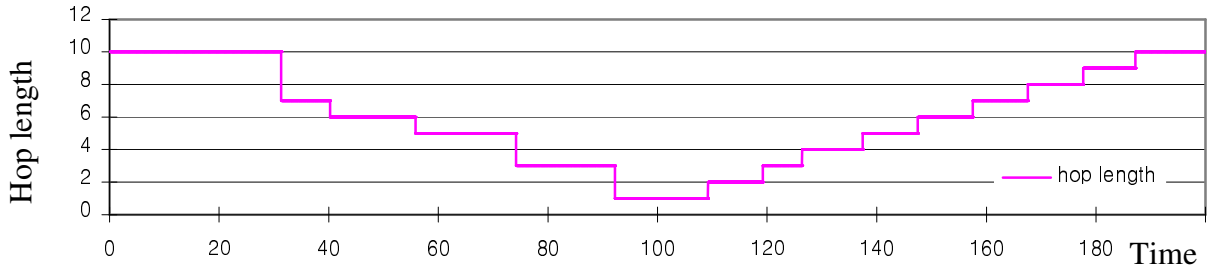
5.3.2 Hop length changes

The following simulations demonstrate how the proposed schemes (i.e., several combinations of the schemes proposed) perform in a given movement pattern and how they affect the routing performance. For observing the change of *adwin* and the subsequent set of *ssthresh*, the *delimiter* did not have an *attenuation factor* and so yielded slightly higher *adwin* than desired. In addition, the sender takes into account the *adwins* of probing packets, rather than a default *modulo4* (due to packet size discrepancy), in order to see the dynamic chane of *ssthresh*. Hence the following simulations represent the fundamental differences among combinations of the schemes proposed. Simulation results denoted in each fugure, such as timeout, fast retransmits, probes, and throughputs, are obtained through a single run of the specific simulation paradigm.

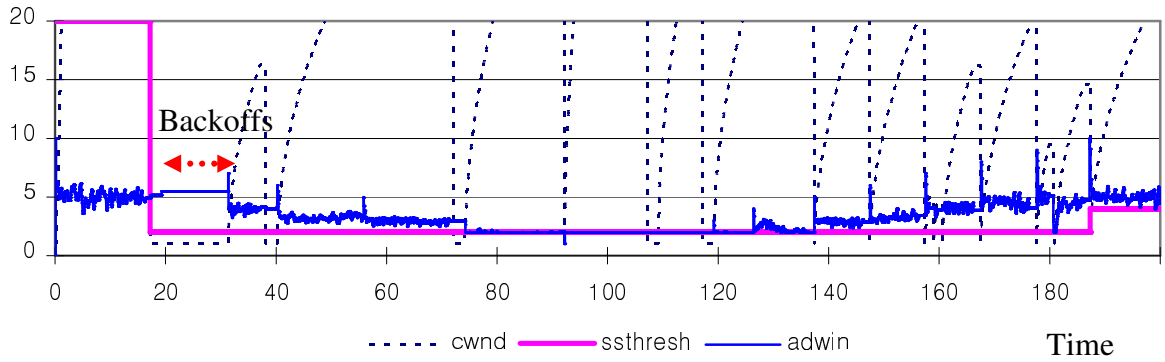
A sender, initiates a FTP transfer of packet size 500 bytes for 200 seconds to transmit to a receiver apart by 10 hop length, each hop 200 m distant (So, maximum hop length variation is at most 2 at once), and at 5 sec, moves towards the receiver in 20 m/s through stationary intermediate routers and then at 95 sec, stops on 10th intermediate router and at 100 sec in 20 m/s, moves back again through to the initial position to arrive at 190 sec.

Given the same movement pattern, the amount of data packets allowed to be injected by the sender affect the amount of routing control packets because on-demand based routing protocol restores the current link whenever it detects a link failure (Chapter 2.4.2.1) that can be caused by overloaded packet injections (MAC collisions). As shown in Figure 5-8 (a), Figure 5-9 (a), Figure 5-10 (a), and Figure 5-11 (a), the controlled sender's window progression by the *delimiter* has a less number of hop length changes than that of Figure 5-12 (a) of the baseline. Immediately after a node gets out of 250 m transmission range, routing protocol invokes the route rediscovery procedure to update hop length. Noticeably, as shown in Figure 5-12, a number of MAC collisions initiate more routing control packets to rediscover a new path link and thus to update hop length more frequently.

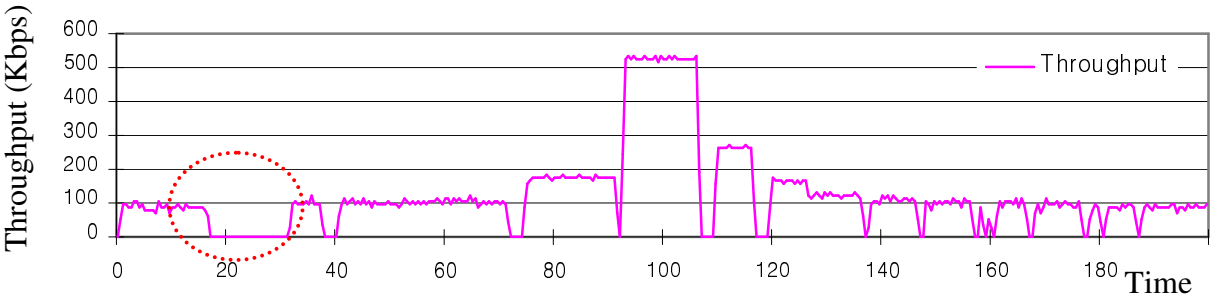
Timeliness of routing control packet flooding to update to find the shortest path link between end correspondents is so important to improve TCP throughput because, as in Figure 5-9 (c) and Figure 5-11 (c), the sender had a longer path link than a possible shortest one due to no invocation of a periodic route update and so had throughput degradation in part; however, undoubtedly, frequent routing control packets flooding might degrade the aggregate throughput though.



(a) Hop length variation, 14 times, here link changes are equivalent with hop length changes. Notice the minimum hop length is 1.



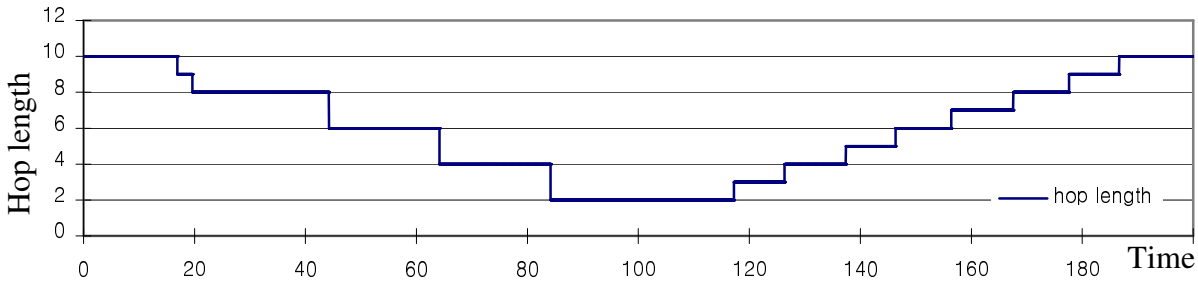
(b) Notice *ssthresh* is almost set to 2 because of the small *adwin*, and backoffs.



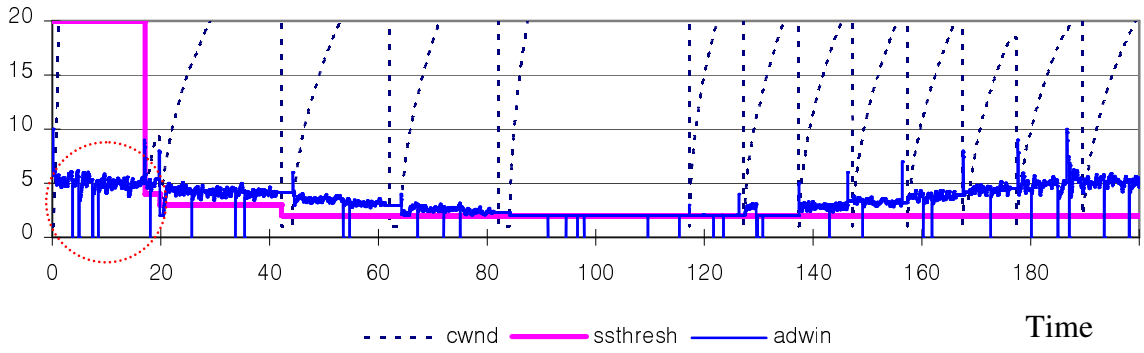
(c) Instantaneously measured throughput with interval 0.1. Notice no transfer by the backoffs.

Figure 5-8. The receiver's delimiter in use. Fast retransmits = 2, timeouts = 20, Throughput = 111.0 Kbps.

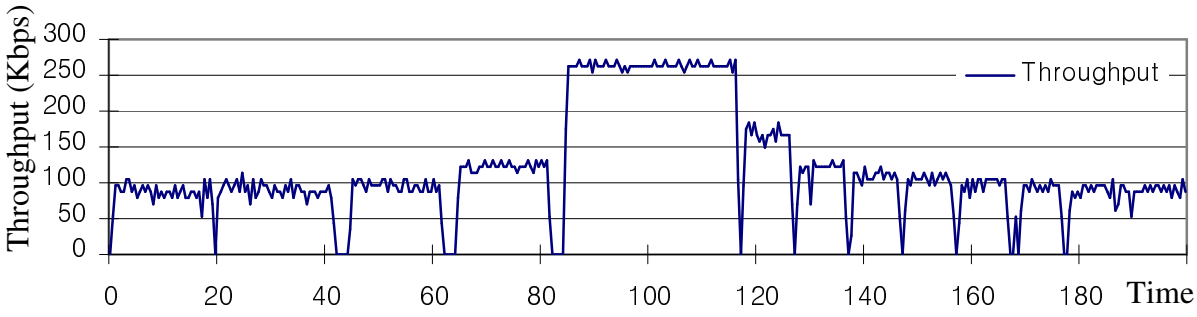
As shown in Figure 5-8, the delimiter seems to throttle the sender's congestion window each time when the path link switches are perceived. In this case, the sender experienced the RTO exponential backoff, as noticed in Figure 5-8 (b), and timeouts almost every time path link was changed.



(a) Hop length changes 13 times. Notice the shortest hop length is 2.



(b) *ssthresh* is set to 2 of the minimum due to the small *adwin* and no enhancements of the sender side. Notice that before 20 sec, a few ZWAs received eliminated the RTO exponential backoffs of Figure 5-8 (b).

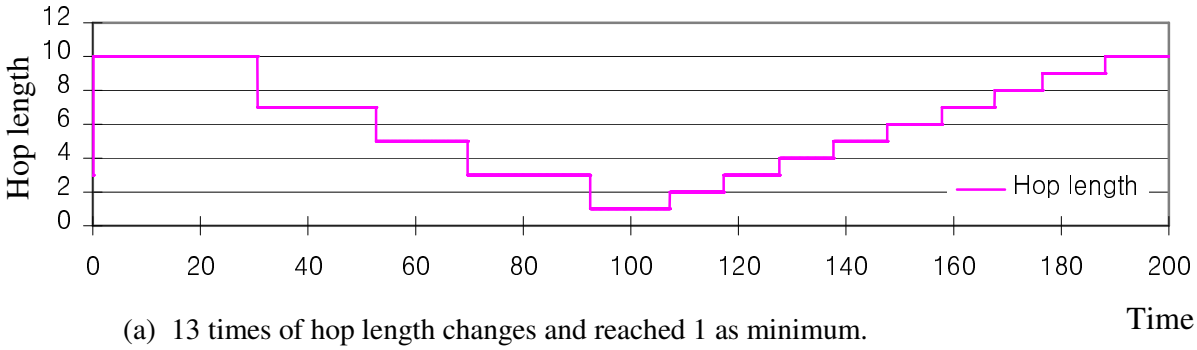


(c) The use of freezer eliminated many timeouts and particularly the exponential backoffs. However, the shortest hop length was 2 (because of less times of routing control packet flooding), and thus throughput was reduced in part when compared in that between 92 and 107 sec in Figure 5-8 (c).

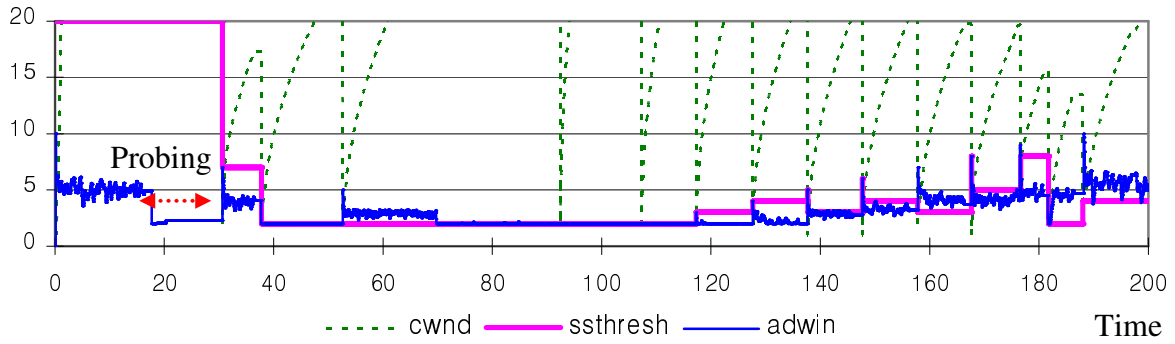
Figure 5-9. The receiver's delimiter and freezer 2 in use. Fast restrans = 3, timeouts = 13, probed = 1, freezing = 53, Throughput in average =106.8 Kbps.

Notice that in Figure 5-8 (b), Figure 5-9 (b), Figure 5-10 (b), and Figure 5-11 (b), all of which employed the *delimiter*, almost every hop link switch caused a timeout. Getting out of the valid transmission range, a MAC sender fails a typical number of MAC attempts (devastating packet forwarding ability) and, by that, reinitiates the routing path rediscovery to maintain the path link. Hence, in the meantime, the TCP sender suffers from the RTO expirations though the path link gets updated to a new path frequently.

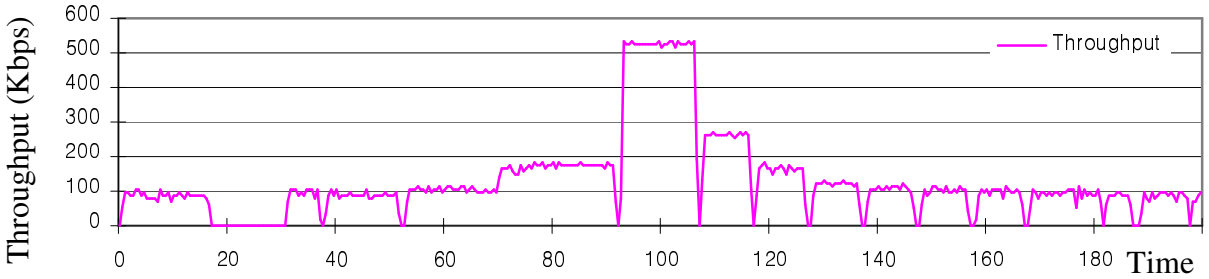
As evident in Figure 5-9 and Figure 5-11, the freezer mechanism reduced the number of timeouts caused by such premature RTO problems to a considerable extent.



(a) 13 times of hop length changes and reached 1 as minimum.

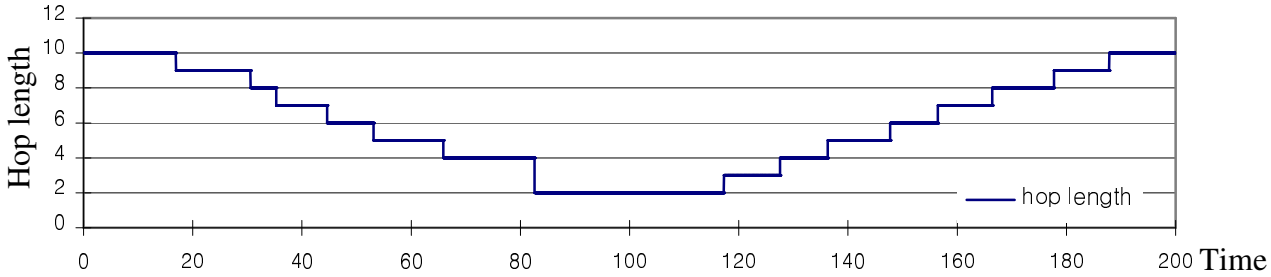


(b) Notice that each time the sender has a packet loss, *ssthresh* is updated by the *adwin* of firstly probed packet after the loss. After the first timeout, the sender has probed from 17 to 30 sec, during which however it suffered from probing backoffs like the RTO backoffs in Figure 5-8 (b).

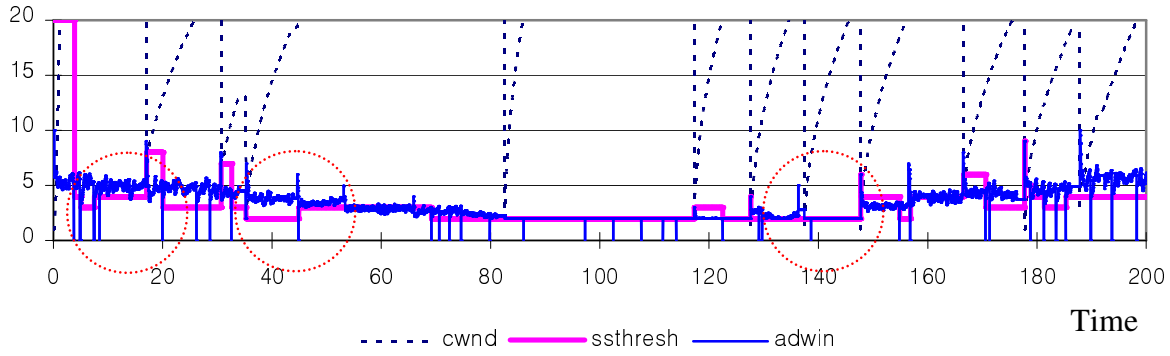


(c) Instantaneout throughput. Notice that by the ad hoc sender enhancements, it had slightly throughput gain in comparison with the throughput of Figure 5-8.

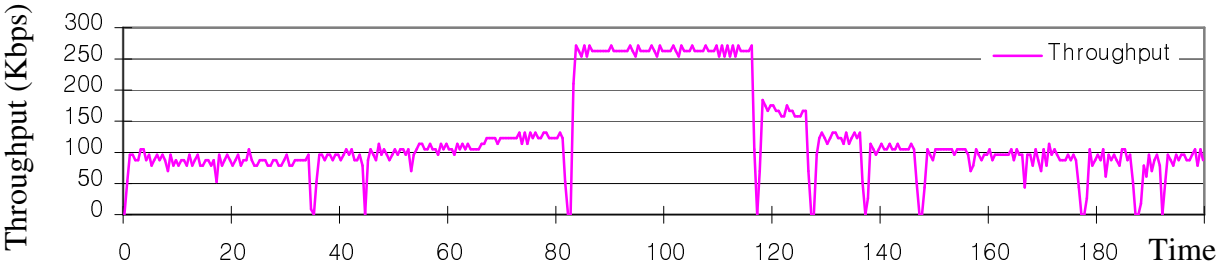
Figure 5-10. The receiver's delimiter and ad hoc sender enhancements in use. Fast retransmits = 2, timeouts 12, probed = 18, Throughput = 116.6 Kbps.



(a) 15 times of hop length changes, Notice that hop length decrements more frequently, but stops at 2 as minimum because of stable transfer.

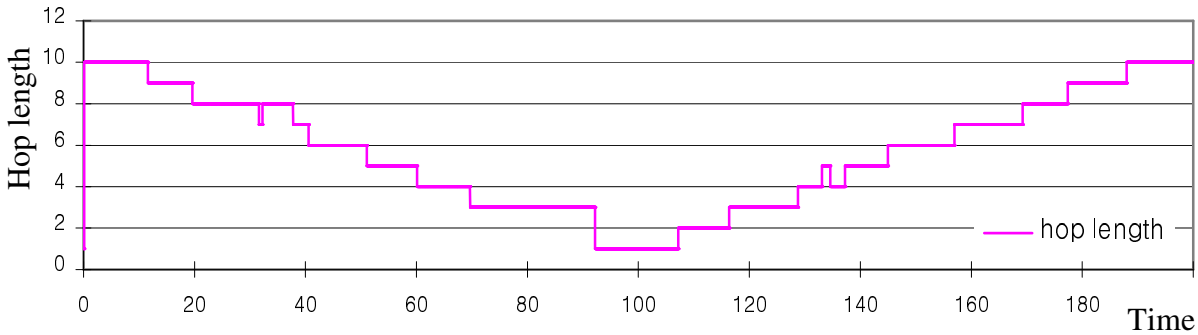


(b) ZWAs by the freezer 2 make the sender update *ssthresh* each time and reduced the number of timeouts. By ZWAs, it calibrates spuriously measured *ssthresh*.

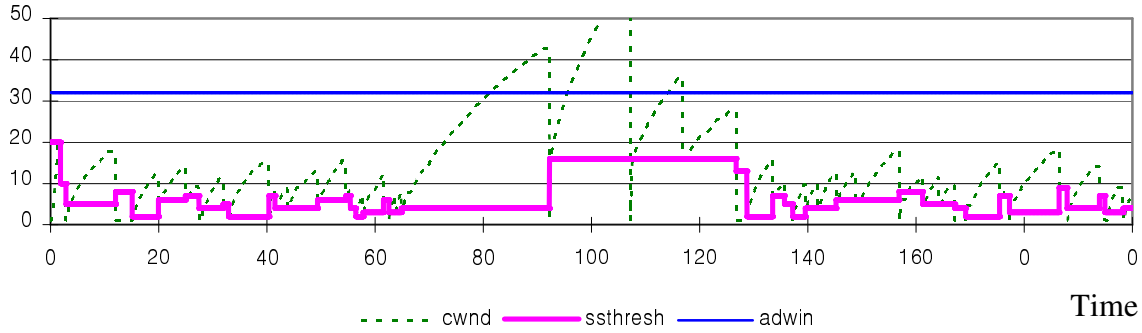


(c) In support of the ad hoc sender enhancements, it had slightly throughput gain in comparison with the throughout of Figure 5-9 (c). However, it has less instantaneous throughput by the shortest hop length of 2 when compared with the peak throughput in Figure 5-10 (c).

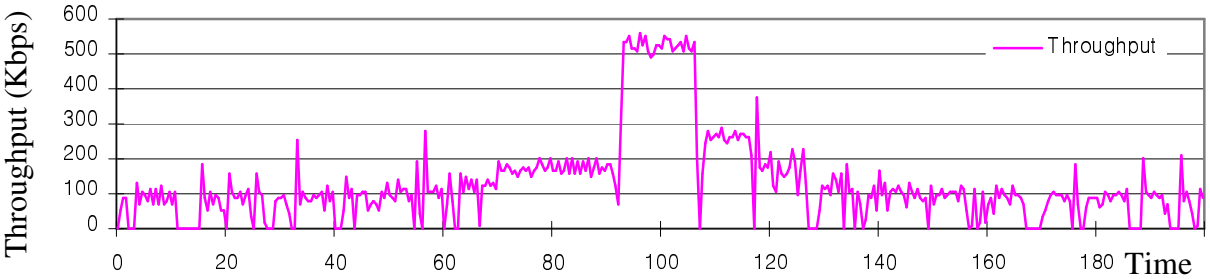
Figure 5-11. Delimiter, ad hoc sender enhancements, and freezer. Fast retransmits = 3, timeouts = 8, probed = 9, freezing ZWAs = 54, and Throughput = 110.3 Kbps.



(a) Hop length changes 21 times. The self collision having more routing overheads flooding to maintains the route frequently requires the need of another path link discovery (In turn, notice the shortest path link was 1), but the network might suffer from the enormous amount of routing overheads.



(b) The sender only controls transmission window and *ssthresh* for which it keeps injecting by incremented *cwnd* until a loss occurs, and therefore the sender likely overwhelms the network capacity and further deteriorates the global fairness in terms of a single common channel to share. See the high *ssthresh* set at 90 to 130 sec of small hop lengths, at which due to resultant fast increasing steady injections, the sender had two timeouts and one fast retransmit during that time.



(c) Instantaneous throughput. A number of losses fluctuated the instantaneous throughput and it implies frequent network capacity overutilization.

Figure 5-12. Base line TCP of adwin set to 32. Fast retransmit =25, timeouts = 20, and Throughput = 117.8 Kbps.

As noticed in Figure 5-12, it obtained a slightly high throughput in total but suffered from a number of timeouts, routing overheads, and, in turn, high medium contention by itself.

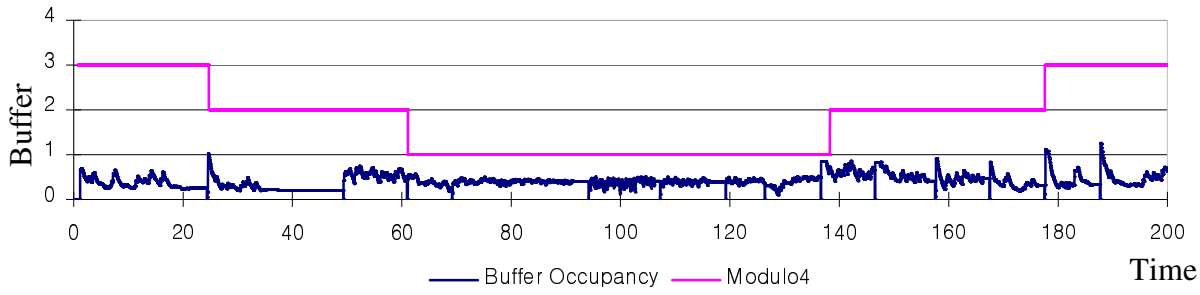
5.3.2.1 Deterministic buffer occupancy

The following shows a newly introduced *metric*, the *buffer occupancy* (i.e., Equation 5-2) obtained when a set of schemes, such as the receiver-only enhancements (i.e., delimiter + freezer), the receiver's enhancements with the ad hoc sender enhancements (i.e., delimiter + freezer + ad hoc sender, namely, *ad hoc TCP*), and the baseline TCP, is deployed for the same simulation performed in Chapter 5.3.2. The *buffer occupancy* for each set was typified through a single run of the simulation, for which the packet size was set to 1000 bytes and the delimiter has specific *attenuation factors* and, once probed, *ssthresh* is set to a *modulo4*. Afterwards, statistical results in

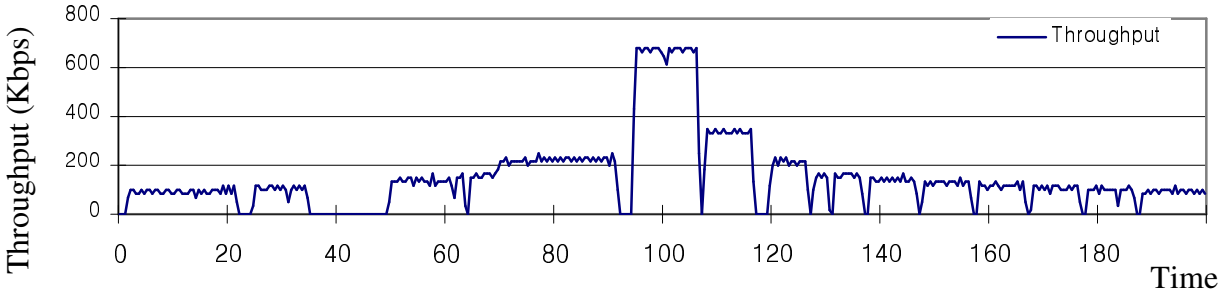
5 % precision are presented in Figure 5-16, where the *ad hoc TCP* had the best goodput and throughput.

As shown in Figure 5-14, the *buffer occupancy* measured for our promising *ad hoc TCP* outperformed the baseline. The baseline has almost fully occupied buffer along the changing upper bound, *modulo4*, over the connection time while the buffer of the *ad hoc TCP* was fluctuating just less than 1 except around 140 sec. At 138.3 sec, a high set of *ssthresh* after probed, because each time probed it is set to a *modulo4*, must have devastated the new path link, even though some time later it was retrieved plausibly by freezing performed to renew the *ssthresh* again. The fluctuation of the buffer occupancy seen in Figure 5-13 (a) and Figure 5-14 (a) implies the presence of medium contention, rather than graceful buffer queuing delay as seen in Figure 5-15 (a), because the delay-sensitive *rater* must relieve buffer queuing to some extent.

The reason why the *buffer occupancy* of Figure 5-13 is less than that of Figure 5-14 is because the *delimiter* worked without the assist of the ad hoc sender enhancements. Each time the sender encountered a packet drop, it halved the current transmission window (in general, *adwin*) for the next *ssthresh*, rather than taking *adwin* itself in the *ad hoc TCP*. Thus, it seems to have a slightly under-utilized window set after each timeout. In addition, as seen in Figure 5-13 (b) around 40 sec, it suffered from the impact of the RTO backoffs. But, no such problem occurred in Figure 5-14 (b), by means of more proper, frequent, updating of *ssthresh* and optimal change of *adwin* complying with the ad hoc sender enhancements.

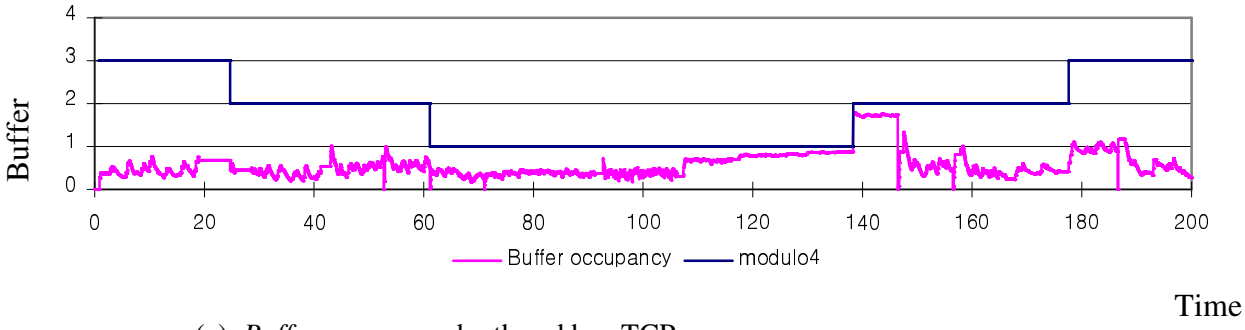


(a) *Buffer occupancy* by the *delimiter* with the freezer

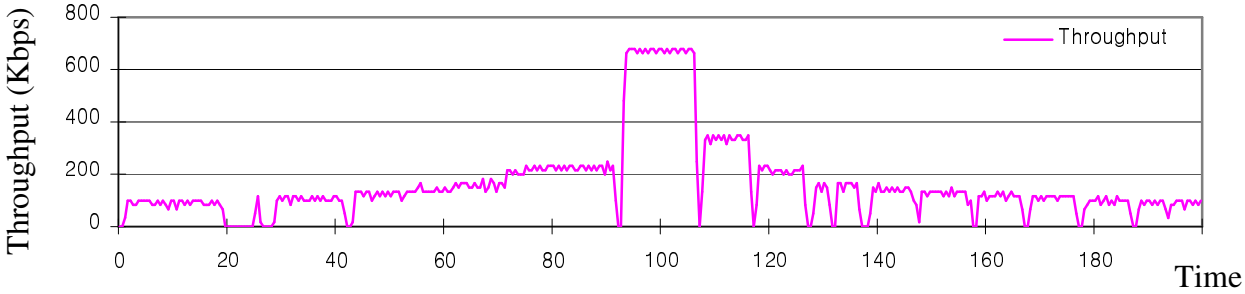


(b) Resultant throughput = 144.9 Kbps, fast retransmits = 0, timeouts = 20.

Figure 5-13. The buffer occupancy according to the receiver enhancements, through a single run.

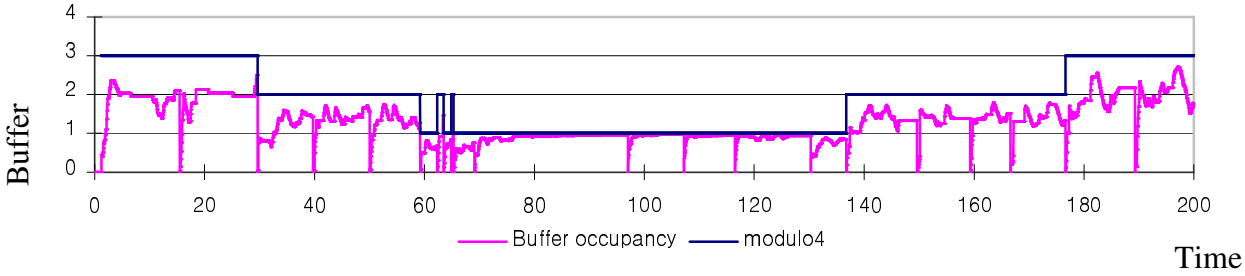


(a) *Buffer occupancy* by the ad hoc TCP

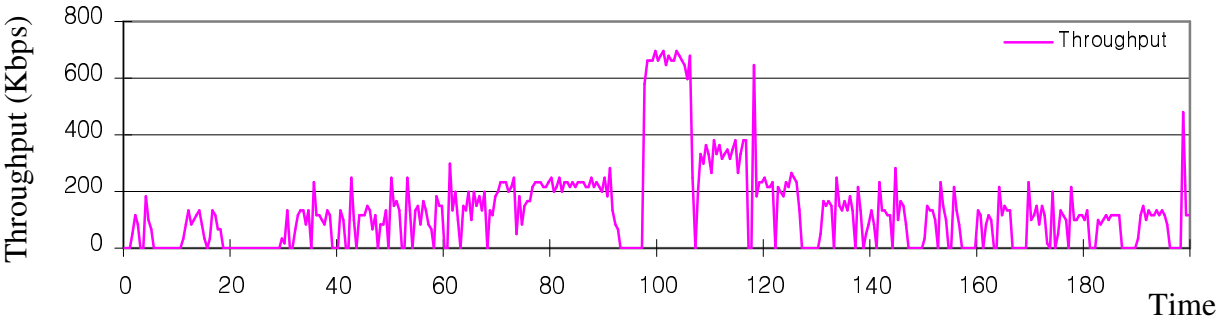


(b) Resultant throughput = 155.2 Kbps, fast retransmits = 0, timeouts = 13.

Figure 5-14. The buffer occupancy according to the ad hoc TCP, through a single run.

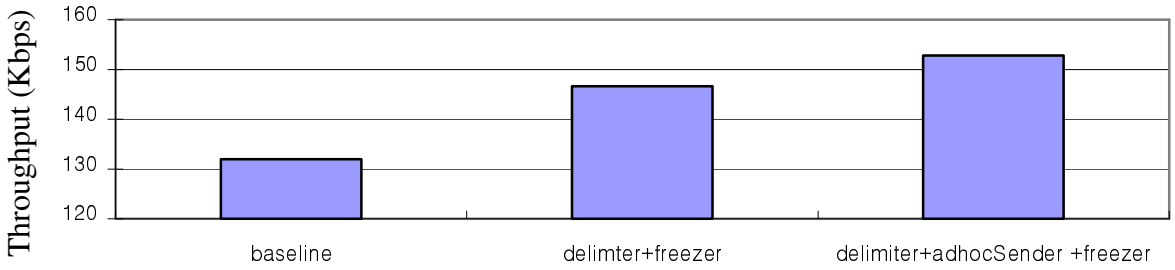


(a) *Buffer occupancy* leveled by the baseline TCP. Notice the large bursts at buffer 1.

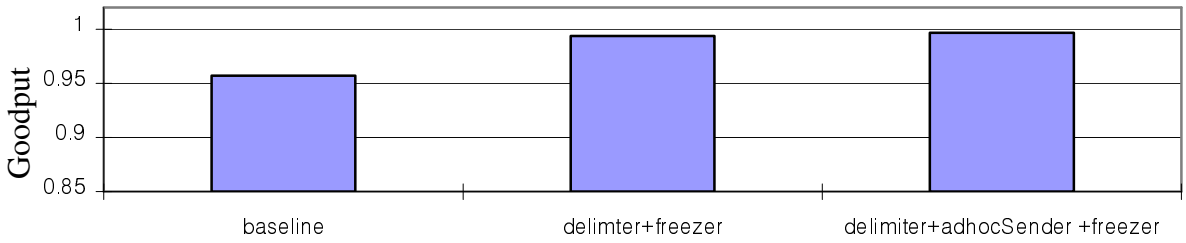


(b) Resultant throughput = 137.3 Kbps, fast retransmits = 25, timeout = 23.

Figure 5-15. The buffer occupancy according to the baseline, through a single run.



(a) Throughput effectiveness among three different schemes.



(b) Goodputs among three different schemes.

Figure 5-16. The throughput and the goodput, (a) and (b), respectively. For baseline, fast retransmits = 23.8 and timeouts = 25.1, for delimiter and freezer, fast retransmits = 0 and timeouts = 17.2, and for delimiter, ad hoc sender and freezer, fast retransmits = 0 and timeouts = 12.6

5.3.3 A single TCP connection in dynamic movement patterns

The following simulation performed now verifies our *ad hoc TCP* for a random movement pattern. Figure 5-17 and Figure 5-18, respectively, shows the throughput and the goodput gain of the *ad hoc TCP* and the delimiter with freezer, comparing with the baseline. Simulation time was 300 seconds. The sender and the receiver were placed and static at the edges of the area, i.e., (100, 150) and

(1400, 150), respectively, of 1500 m x 300 m retangular flat, which forces lengthy chain. A FTP was initiated at 0 sec and lasted at 300 sec. Additional 50 mobile nodes are moving in the area in order not to suffer from the network partitioning during the communication.

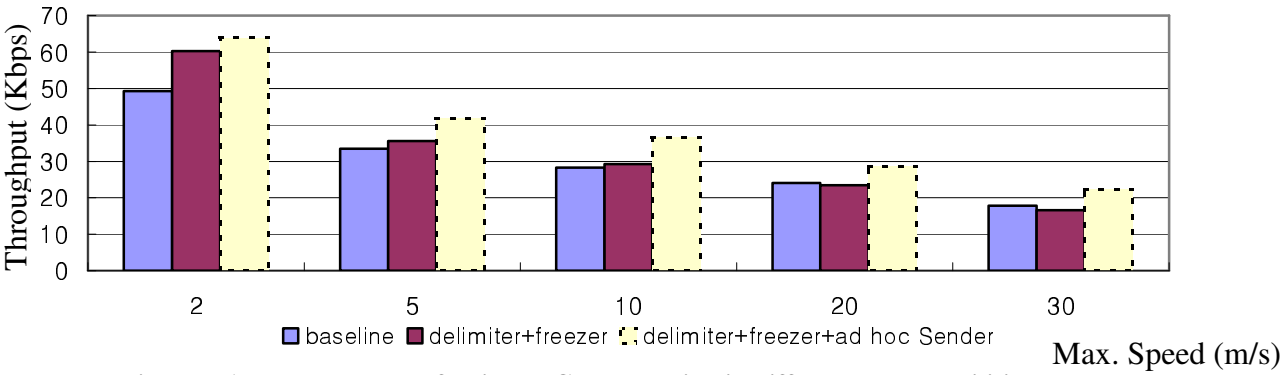


Figure 5-17. Throughputs of a singe TCP connection in different node mobilities.

As shown in Figure 5-17, the *ad hoc TCP* outperformed the others. When nodes are moving so fast (as in 20 m/s, and 30 m/s), noticeably, the receiver-only enhancements (i.e., delimiter + freezer) obtained reduced throughput than that of the baseline. Due to the absence of the ad hoc sender enhancements, the TCP sender is still likely to suffer from the RTO backoffs even though the receiver end enhancements are effective as evident in the stationary situations. The reasoning behind this is that in the higher node mobility, relatively fewer in-flights resided in the *pipeline*, compared to the baseline, can cause the sender to suffer from the RTO backoffs due to the likelihood of starving *pipeline* (i.e., no in-flights). Furthermore, the sender cannot guarantee the timely, reliable delivery of ACK, as well as ZWA if advertised, which results in further performance degradation.

However, the receiver-only enhancements gained a better *goodput* than the baseline, which justifies its beneficial use in the dynamic movement patterns as well. Most promisingly, the *ad hoc TCP* had 30 % throughout gain, and 15 % goodput gain as seen in Figure 5-18.

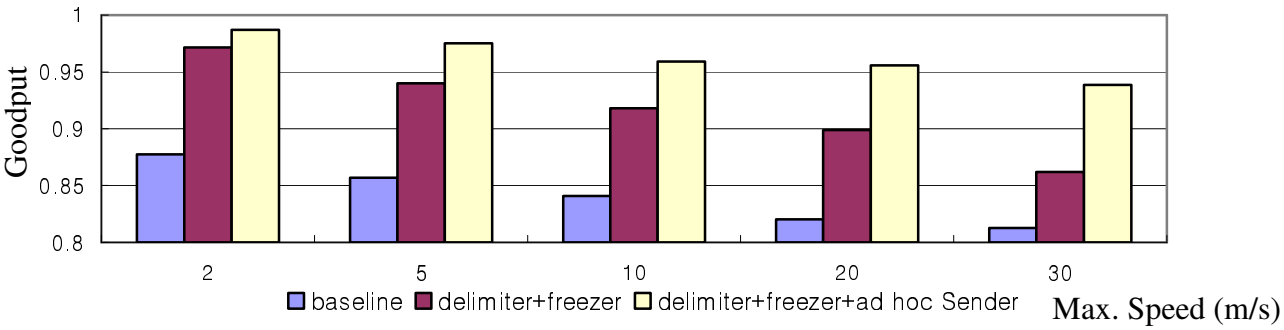


Figure 5-18. Goodputs of a single TCP flow in different node mobilities.

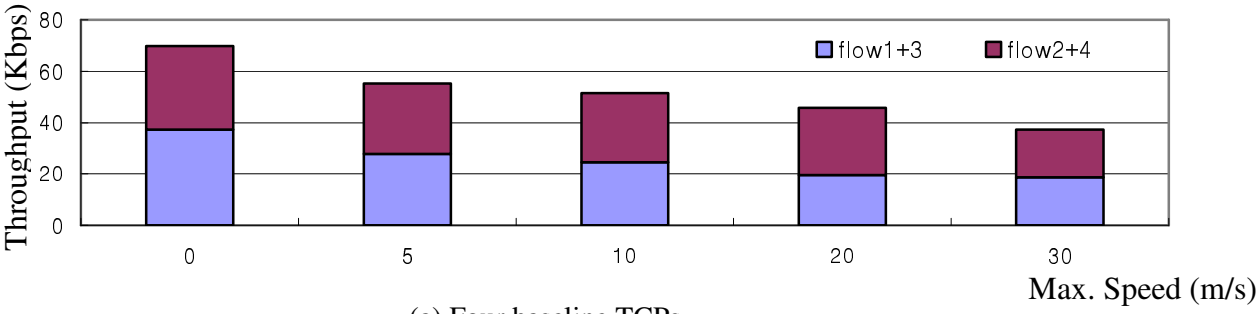
The baseline has been told that it is likely to suffer from the large bursts of data due to the reative change of the transmission window. As a negative paradigm of the baseline, the shorter RTT flows have fewer timeouts and a larger congestion window causing them to contend more aggressively (e.g., Figure 5-15 (a)), while the longer RTT flows are more likely to backoff and reduce their congestion window. Such bursts of the short range flows might hamper other neighbors' transmissions. It then comes to unexpected forwarding blocking that degrades fairness among concurrent flows, as well as causes substantial routing overheads. In turn, a mechanism based on a rate control attempting at avoiding the bursts of data would be more suited for multi-hop interfering ad hoc environments [104]. In the sense that the *delimiter* proactively limits the sender by means of the employed *rater*, the following Chapter will address the fairness provided by the *ad hoc TCP* and evaluate the friendliness between the baseline and the *ad hoc TCP*.

5.3.4 Multiple connections and friendliness among TCP flows

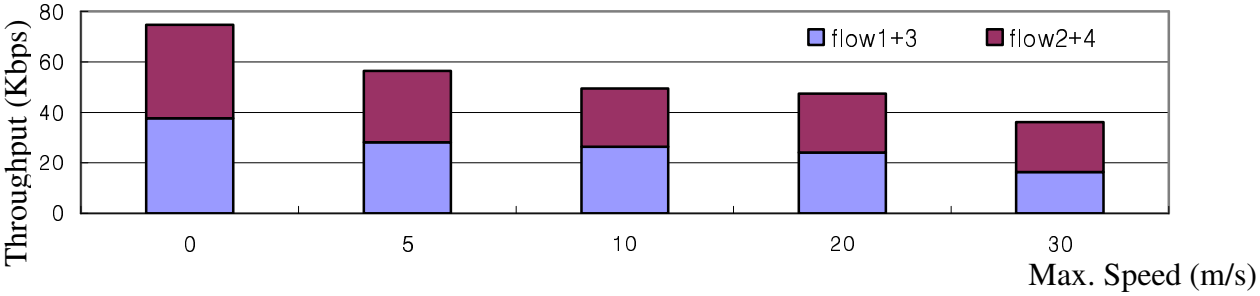
The following simulations show the performance of our *ad hoc TCP* in the network where there are competing multiple connections of homogeneous TCP pairs of each of the baseline, *ad hoc TCP*, and Newreno TCP and of mixed pairs of *ad hoc TCP* and Newreno TCP (as most widely deployed) to see the friendliness in between. Likewise in Chapter 5.3.3, to avoid the event of destination-unreachable situation in terms of different node mobilities, 50 additional mobile nodes are added and mobile in the area of 1500 m x 300 m rectangle (each node with the max. speed but zero pause time for continuous movement). The communicating mobile pairs are positioned at the edges of the network topology and static, so that they share the bandwidth with each other acoss same bottleneck area formed between end nodes, which is similar to the simulation performed at [58]. In a given mobility pattern, accordingly the constellation of mobile nodes between end nodes is mobile.

5.2.2.1 Homogeneous TCP flows

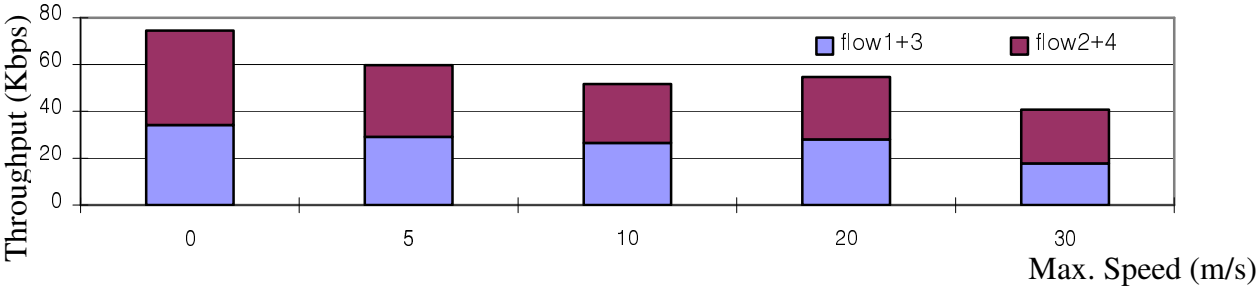
Figure 5-18 shows the TCP throughputs obtained when four TCP flows of identical TCPs were present. Not surprisingly, the *ad hoc TCP* outperformed other TCP sets.



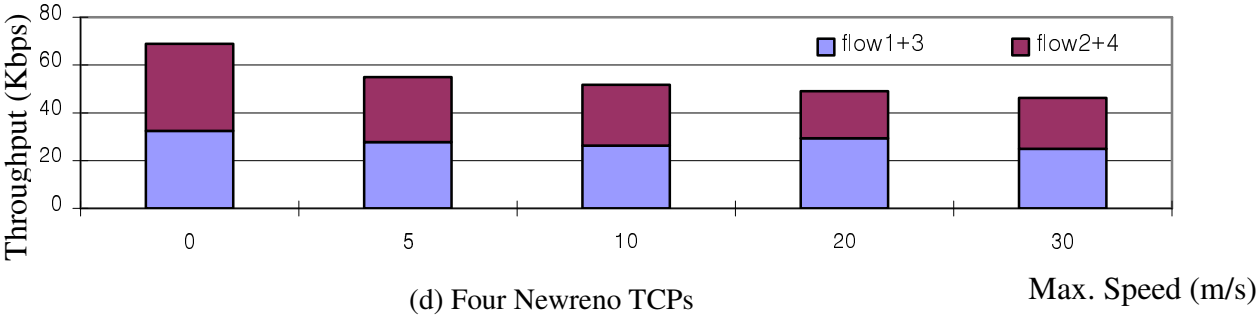
(a) Four baseline TCPs



(b) Four TCPs of delimiter + freezer



(c) Four *ad hoc TCPs*



(d) Four Newreno TCPs

Figure 5-19. Aggregate TCP throughputs in changes of max. node mobility.

Interestingly, at 30 m/s in Figure 5-19 (d), the set of Newreno TCPs obtained a slightly higher throughput than that of the *ad hoc TCPs*. Newreno TCP must be less aggressive than the baseline in terms of loss receovery as seen at 20 and 30 m/s for which it outperformed the baseline (see Figure 8-5 (d) and, particularly, (f)), probing further bandwidth. However, unlike the *ad hoc TCP*, the baseline, and its variant, Newreno, that both reactively shirnk their transmission window, are likely to cause large bursts of data and, thus, increasing buffer utilization causing RTT increase. Subsequent RTO inflation results in relatively longer timeouts, so they are more likely to suffer from the long RTO backoffs rather than the *ad hoc TCP*. Yet, such trend of large RTOs helps with attaining the friendliness with the *ad hoc TCPs*, as evident in the following Section, to the extent that in the meantime the baseline suffers from the large RTOs, the *ad hoc TCP*, whose transmission window is eventually reduced to 2 (as a predefined W_{LOWER}) due to the aggressive Newreno, can transmit.

The most prominent merit of use of our ad hoc TCP is that as showin in Figure 5-20, the ad hoc TCPs promise the best *goodput* in comparison with the baseline and Newreno TCPs (up to 22 %). Figure 7-5 in Appendix A.8 shows the individual *goodputs* of the four TCP flows according to the different max. speeds.

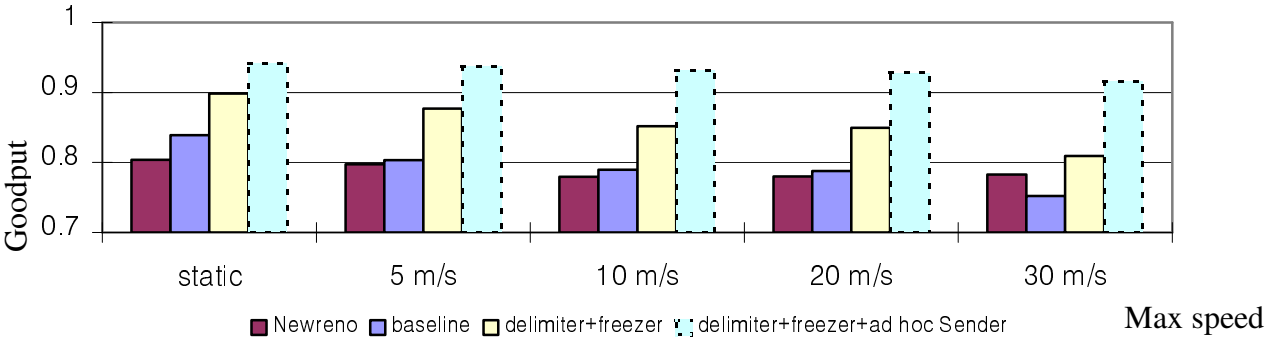


Figure 5-20. Goodputs of the aggregate of four TCP flows in mobility applied. Max. 22 % goodput gain at 30 m/s. A number of probing packets flooded (especially for the *ad hoc TCPs*) after each frozen were not taken into account.

5.2.2.2 Disparate TCP flows

The following example applied four TCP flows but two pairs of Newreno TCP and of the *ad hoc TCP*. As shown in Figure 5-21 competing with each different type of TCP, to some extent in comparison with Figure 5-19 (c), the fair share of medium between the *ad hoc TCP* and Newreno flows seems to be reasonably achieved staying pretty much unchanged (still shared almost equally as in Figure 5-19) except that for high node mobility, it slightly stole some bandwidth from that of

the *ad hoc TCPs*. As mobility goes higher, the pair of Newreno steals some portion of bandwidth from that of the *ad hoc TCP*. The *ad hoc TCPs* suffered from a number of probes because relatively larger bursts cause by Newreno TCPs, than the ad hoc TCPs, deteriorated the medium availability. As compared with Figure 5-19 (d) of Newreno TCPs, slight throughput improvements were appeared at high mobility.

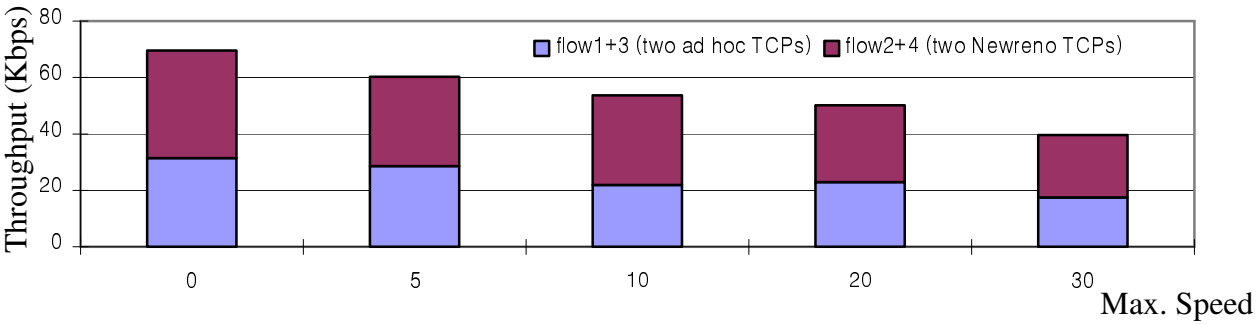


Figure 5-21. Friendliness of disparate multiple TCP flows.

As a result, with respect to fairness provided by transport protocols, different TCPs being present convey different link utilization. As evident in Figure 5-19 (c) and (d) and Figure 5-21, the available medium cannot be fully utilized when compared to the identical set of TCP, nor guarantee fast packet forwarding, suffering from the hardship of an optimized time scheduling. It requires a controlled use of medium in an identical manner to transmit packets at the sender.

Tolerably, Newreno TCPs behaves friendly with the *ad hoc TCPs*, but as a matter of fact a bit aggressive window evolution of the reactive nature adversely affects the global fairness achievement in fair uses of a single common channel among all other identical competitors. As a consequence, it is acceptable to an extent that the *ad hoc TCP* co-exists with the greedy reactive TCP in perspective of throughput, but not in *goodput.*

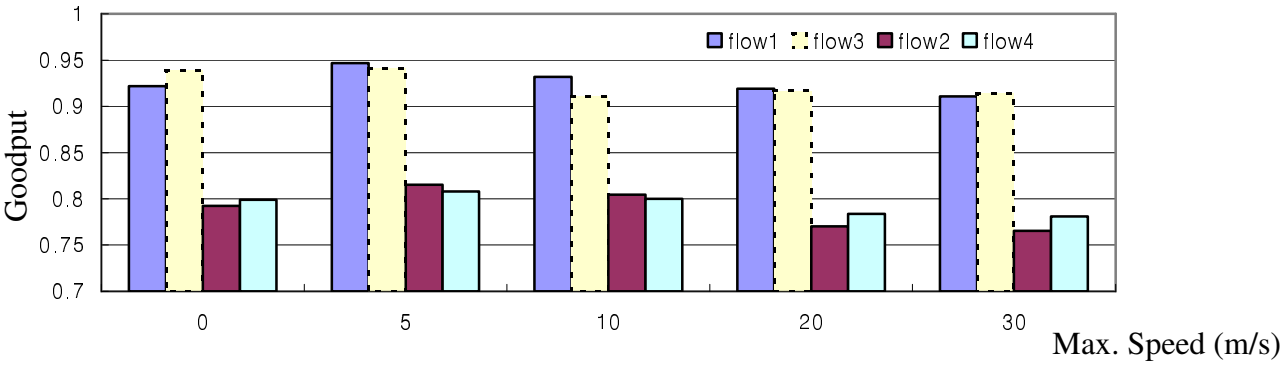


Figure 5-22. Goodputs of disparate multiple TCP flows. Where flow 1 and 3 are ad hoc TCPs and 2 and 4 are Newreno TCPs.

Ever prominently, Figure 5-22 shows that the *goodput* of the *ad hoc TCPs* was superior to that of Newreno TCPs, up to 19.4 %.

5.3.5 Hop length unchanged but path link switched

The inspection of TTL value every receiving data packets may not be able to signify of all hop link switches, but informs how many hops recently arriving data packet has travesed through. If a number of fast mobile nodes are in a small area, the path link changes so frequently even though the hop length does not change. The following simulation performed verified that even though a different route but the hop length is still maintained, the *rater* can converge to the optimum sooner or later.

To verify the fast convergence to a new route condition in case of path switches with the same hop length (where all the information, i.e., SFLD, FLDV, FLDD, contention factor, and freezing timer, measured at the receiver will not be renewed but still in use), a simple experiment was performed. There were 50 nodes moving in several different max. speeds, 10, 20, and 30 m/s with zero pause time for continuous moving in the square toplogy of 1000 m x 1000 m for 300 seconds. A randomly chosen pair of the sender and the receiver encountered a number of path link changes but in the same hop length because the mobile nodes likely gather around the center of the square topology with the random way point mobility pattern. Thus, hop length between end nodes was eventually ended up to short delivery paths (i.e., average hop length of 5 at 10 m/s, 4 at 20 m/s, and 3 at 30 m/s).

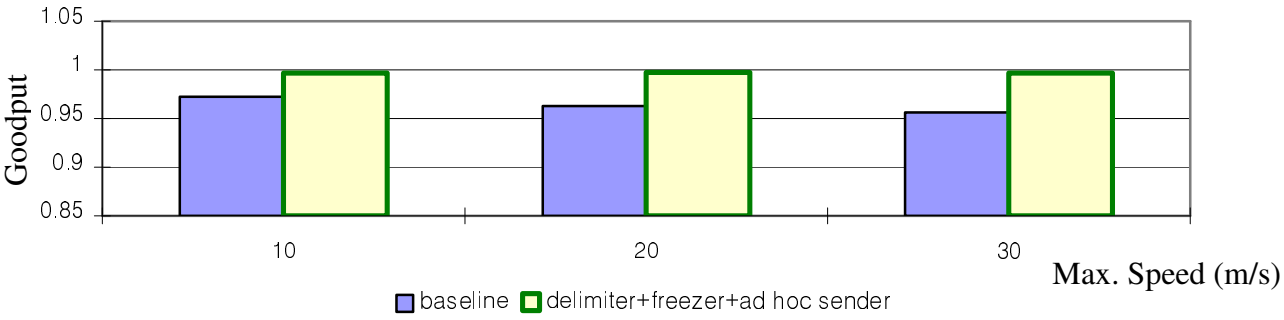


Figure 5-23. Measured goodput in the square topology.

Figure 5-23 shows still outstanding *goodputs* measured for the *ad hoc TCP* and the throughput improved up to 31 % at max. speed 30 m/s. The baseline suffered from medium contention related

packet losses and high buffer queue-up, as evident in Section 5.3.2.1. The throughput gains according to mobility changes are found in Appendix A.6.

5.3.5.1 Use of short-lived connections

Due to diverse traffic types such as short-lived interactive (i.e., telnet) and transactional traffic (i.e., HTTP) as well as bulk data transfer (i.e., FTP), the rate controller might not work properly to show its beneficial use because of its shortness of connection time that is often too short to get a sufficient knowledge of the network condition. From Section 5.3.2 and Figure 5-23, looking-fast convergence in hop link changes and sustaining promising *goodputs* (even though the receiver cannot renew the measurement for computing *adwin* and other contingent measurements because of the different path link of the same hop length) confirms the use for the transient short-living connections.

Chapter 6
CONCLUSION

6.1 Summary

In this thesis, we have verified the merit of use of our elaborated TCP schemes outperforming the baseline and Newreno TCP. In the early stage, the *rater* using the *scale factor*, *1/n*, has performed promisingly over different hop lengths, as seen in Table 5-2 and Figure 5-1, and in support of the freezer employed, as in Figure 5-7, and the *delimiter* using the *attenuation factor*, *modulo4/n* characterized by 802.11 (Figure 3-14), gave a bit more aggressive limitation of the *adwin* (Section 3.4.7.3) than the actual optimum, as noticed in Figure 3-13, thus demonstrated slight underutilization relatively, as in Figure 5-2, but ever expressed superiority over the baseline in g*oodput*. The more strict analysis of utmost medium availability in 802.11 will derive more reasonable *attenuation factors* to apply.

The freezing timer is sensitive to the change of β in terms of freezing frequency, but has been verified it is quite useful to timely freeze the sender and give a chance to renew *ssthresh*, obtaining considerable gains as in Figure 5-3 and Figure 5-4. Moreover in support of the *ad hoc sender enhancements*, subsequent probing after frozen differentiates the after-behaviors beneficially in comparison with the receiver-only enhancements, as evident in Figure 5-5 and Figure 5-6.

As a result, the consistency in *goodput* over different mobilities encourages using *ad hoc TCP* through the network-wide deployments, instead of conventional reactive TCPs; in high mobility the *ad hoc TCP* obtained considerable gains both in throughput (up to about 30 %) and especially *goodput* (up to about 20 %) in spite of the expense of additional bandwidth consumption for the add-on probing capability.

As supplementary approaches, *Buffer occupancy* and *contention factor* has been evaluated for its beneficial uses in terms of the network buffer dimensioning and the contention degree determination, respectively, and further addressed in Appendix A.5.

Furthermore, the prototype of use of the receiver-oriented flow control encourages further applicability into the wireline end TCPs because of its easier deployability, confirmed by only receiver-end modifications, and compatibility, by using the receiver's window advertement, when supposing that a plausible *attenuation factor* is available.

The next Chapter envisions future directions with some issues comtemplated for further research.

6.2 Future work

- As a practical point of view, we are further encouraged to evaluate the performance of the elaborated pair of TCP when other types of background traffic patterns are applied.

- As aforesaid, we should further study how to choose and change α and β of the contention factor and the freezing timer, respectively, if necessary in term of change of the network *reference metrics.*

- Optimum probing mechanism in terms of probing timer backoff will be required to have not not exponential but adaptable probing interval, so it will not inflate RTO, somehow, by means of taking into account the link condition (e.g., whether the hop length has been changed or not) after the probing is succeeded. The awareness whether the link has been changed might give further information to confirm the validity of timestamp information of probed packet. Either sender or receiver may change the timestamp of the probe in order to make it adaptable to the currently available link and so not to inflate the RTO. In addition, the use of smaller probe than data packet does not quite inflate the RTO and relatively does not impose substantial bandwidth expense. However, it still requires a better translation to validate the RTT information of the small probing packets.

- The sender's historical measurement of SRTT, RTTVAR, and, in turn, RTO must be revised for use in ad hoc networks, where there are frequent route changes and so the values become stale. Thus those measurements should be renewed or changed each time it perceives the current link changes—it requires the cross layer information flow as if the receiver uses the TTL values of incoming data packets (i.e., SFLD, FLDD, and other terms are renewed each time link changes). Similarly, SRTT and RTTVAR can be renewed each time hop length changes, in support of the receiver's perception of the hop length change.

- More elaborate sender's congestion window progression strategy is required because the current sender can only either increment one packet per an incoming ACK or per RTT, in the slow start phase and the congestion avoidance phase, respectively. That is, it should have a more dynamic window progression way, i.e., switched from/to either phase according to the fast changing link condition (perceived by the sender, or the receiver that utilizes either the *buffer occupancy* or *contention factor*) instead of perceived, and then switched, by an actual packet drop.

- Beneficially, the receiver's enhancements can compatibly make a control of a wireline sender, which further requires an optimum calibration strategy in advertisement of correct *adwin* to give an effective delimitation because the initial restriction of the current *delimiter* set might be likely to be too conservative due to the combination of ad hoc and fast wireline links (given that no buffer congestion is in the wireline network). Further study is required so that it can find a bit aggressive *delimiter* adjustment by investigating a proper *attenuation factor* to perform promisingly.

- To proliferate the commercial use of ad hoc networks in public, regionally pre-existent routers to provide supervised and centralized services, and also to perform reliable packet forwarding services, could be prepared. In case, more dedicated routers coordinate the requests of receivers that would pay for dedicated services, such as robust forwarding capability and intelligent network-wide feedbacks.

- Today, there exist many different transmission mediums and intermediate routing policies between end correspondents. Therefore, packet loss itself no longer need the reduction of the transmission window, rather requires to clarify the available throughput to set a plausible slow start threshold. So, the end nodes should behave, not relying on any network-originated beacons, but rather on vertical information crossover between layered protocols (if the cross-layer information flow provides more accurate and useful information by the credibility of each layered protocol). For example, an end node can be aware of what type of its end opponent or what type of network it resides in or what kind of medium data traverses through, by means of a passive discrimination way or the use of an IP option field. Then, the sender can control transmission rate accordingly, as well as the receiver can properly coordinate the *attenuation factor*.

Chapter 7
APPENDIX

Appendix A.

A.1 The passive clock-synchronization at the receiver

If no synchronization is attained between end hosts, we should use a passive clock synchronization strategy as proposed here so that it could determine the clock offset and is then able to give a coarse but meaningful forward link delay.

H. Jiang and C. Dovrolis [79] proposed and evaluated a passive measurement methodology of RTT so that it can monitor it in the middle of network (i.e., monitoring router) for particular purposes, such as required buffer dimensioning. Extending the simple principle to be coupled with the beneficial use of the FLD aforementioned, the receiver can estimate RTT and, by halving the RTT measured, give a coarse FLD but meaningful. Of course, in general, the forward link delay estimated could be underestimated because the packet size of data is larger than of ACK. In case, if forward link delay is measured smaller than actual, the initial rate estimated goes to be higher than the actual, and vice versa.

Specifically, in early connection establishment periods like the 3-way handshake, a RTT is initially acquired by the time taken between one packet (SYN) sent from the receiver and responded packet (SYN-ACK) received from the sender (i.e, passive open), (or from an active open case), and by half, the coarse[1] but meaningful FLD is obtained, which gives the clock offset value. It might be validated by a certain ratio where for example if the clock offset is within a certain portion (i.e., threshold to validate in comparison with actual opponent timestamp) of the computed FLD, we should discard the estimated one but take the current opponent clock because it is tolerable to use directly and otherwise, if outside the threshold, take the coarse clock offset estimated.

[1] In general, the FLD is greater than the Reverse Link Delay (RLD) due to larger data than ACK, and so the FLD seems to be overestimated, which gives the initial network bound capacity to be underestimated.

Furthermore, the receiver may calibrate, or validate, the clock offset by validating the ratio of changing FLD and FLDV, measured by every subsequently arriving ACKs, in comparison with plausible *reference metrics* in terms of the number of hops, packet size, moderate network condition, etc.

A.2 Stationary string toplogy in change of hop lengths

Table 7-1 shows the merit of use of any combination of our propositions when compared to the normal TCP SACK of Table 5-2.

Hop length				5		10		15		20		25		30	
delimiter		Throughput (Kbps)		128.5		77.9		77.8		63.7		54.9		52.6	
		Fast retran	timeout	0	2.2	0	4	0	2.5	0.1	4.9	0	7.4	0.4	7.1
delimiter+AdhocSender		Throughput		128.6		84.7		80.11		67.5		54.8		49.3	
		Fast retran	timeout	0	1.4	0	0.8	0	0.6	0.2	2	2.1	4.3	0.2	3.6
delimiter+ Freezer+ AdhocSender	Beta3	Throughput		126.4		82.4		71.5		71.1		53.5		53.0	
		Fast retran	timeout	0	1.8	0	0.4	0	1.8	0.2	2	0.5	3.8	0.4	3.5
	Beta5	Throughput		128.4		84.1		76.2		64.3		55.4		52.0	
		Fast retran	timeout	0	1.6	0	0.7	0	0.8	0	2.3	0.5	3.2	0.6	3.3
	Beta7	Throughput		129.2		84.2		68.2		62.2		60.8		47.2	
		Fast retran	timeout	0	1.2	0	0.8	0	2	0	2.5	0.7	3.3	0.1	3.3

Table 7-1. Simulation results when employing the delimiter, the freezer (in change of beta), and the ad hoc sender enhancements (Figure 5-5). Experimented in stationary string paths, packet size 1000 bytes, 100 sec simulation time. The ad hoc TCP seems most outstanding.

A.3 Passive Smax measurement in the network simulator

In ns given configurations tabulated in Table 8-2, the maximum throughput (Smax) was measured over one hop communication where no other contending neighbor was present, supposing that the amount of MAC overheads is identically applied irrespective of the packet size.

Parameter	Value
TCP Packet size (MSS)	500 bytes
ACK Packet size	40 bytes
TCP/IP header	40 bytes
Routing overhead	20 bytes
MAC overhead	62 bytes
Raw line rate	2 Mbps
Measured RTT per hop (FLD = 0.00538)	0.009 s (in average)

Table 7-2. Simulation parameters and measured RTT per one hop

Using the measured RTT per hop in average from Table 8-2, the minimum MAC overheads take up in time 0.00347 second regardless of packet size (this may be explicitly computed from the overheads including IFS intervals against the inaccuracy of measurement due to other processing delays), which gives Smax (726.43 Kbps) and with which, from Equation 3-2, Smax changes according to varying packet size as shown in Figure 8-1.

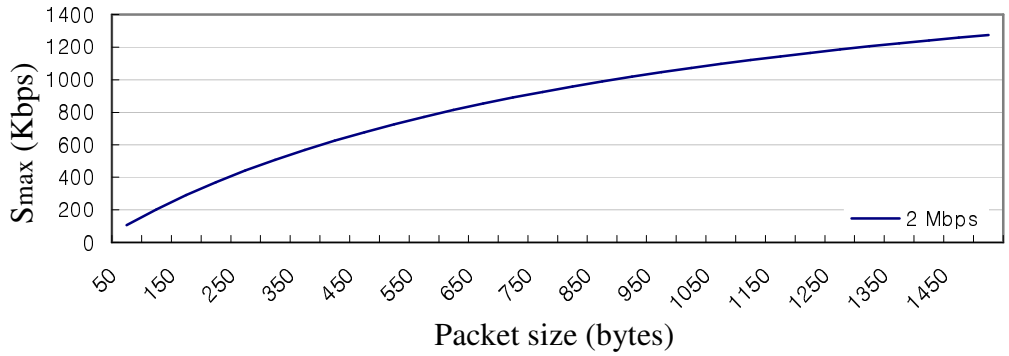


Figure 7-1. Varying Smax in change of packet size with the given minimum MAC related delay.

A.3.1 Empirical perception of Smax by aid of confirmed ad hoc senders

If the ad hoc receiver cannot make a sure of Smax due to unspecified amount of lower layers overheads[1], a passive measurement can also be performed to give Smax value empirically over all TCP connections the receiver has made with ad hoc senders being confirmed by a single bit notification. Smax will be computed when the receiver experiences a minimum FLD, which gives the best Saverage according to the hop length, as a *reference metric* of path link identity.

A.4 Communication with high speed wired sender

In principle, when an ad hoc receiver communicates with a wired high speed wireline counterpart, the receiver generally overestimates the bottleneck throughput because of the end-to-end computation of FLD and the number of hops and thus cannot restrict the sender's congestion

[1] $S_{\max} = \frac{MSS}{MSS + Total\_\min\_overheads} \times B_{raw}$, where *Braw* is 2 Mbps. Smax is thus easily determined at W=1 (no contention) by $n \times MSS / FLD$.

window progression, which may harm medium constraint network path because it progresses until it encounters a packet loss.

Beneficially, from Equation 3-9, as alluded in Chapter 3.4.6.1, if $S_{average}$ is greater than $S_{\max}$ (confined by identical ad hoc link maximum capacity, less than 2 Mbps if with overheads), the receiver at that moment can verify its opponent resides in (say) wired high speed network that has at least more than one hop link whose capacity is greater than the ad hoc one. In response, it can intelligently take a bit more aggressive (less than computed) advertisement by employing a different *attenuation factor* to be able to properly delimit the sender.

To advertise reasonable restricted window, the receiver should keep in mind the number of bandwidth-limited ad hoc hops along the path in corporation of base station. Base station that gives a receiver access to the wired sender should inform the receiver of how many hops of the ad hoc links packets traverse through, probably in support of the timestamp option. Resultantly, it can give a more accurate $S_{average}$ that is now only taking into account ad hoc links. Further study could be required in order to find an optimum calibration to give a better initial restriction.

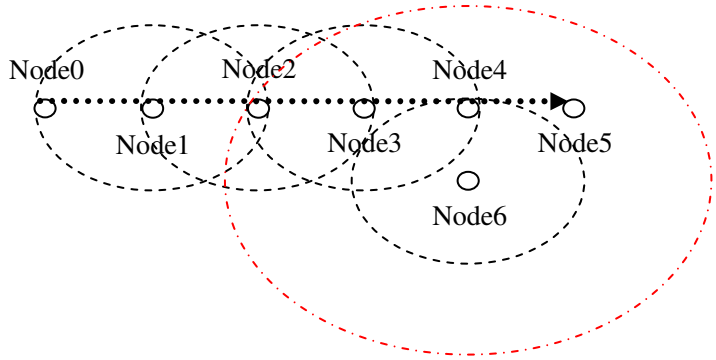
In addition, just in case, in order to prevent the adverse impact of built-in queuing level in the wireline routers because there are many other types of aggressive sender TCPs that usually do not take into account the sensitivity of packet traveling time, the deterministic *buffer occupancy* will be required and probably makes the *delimiter* and the freezer work only when exceeding a threshold of the *buffer occupancy*. So, the receiver accounts for the current *buffer occupancy* of the path link in order to limit the wireline sender as if TCP Vegas takes two thresholds representing buffer size.

A.5 Buffer Occupancy and Contention Factor

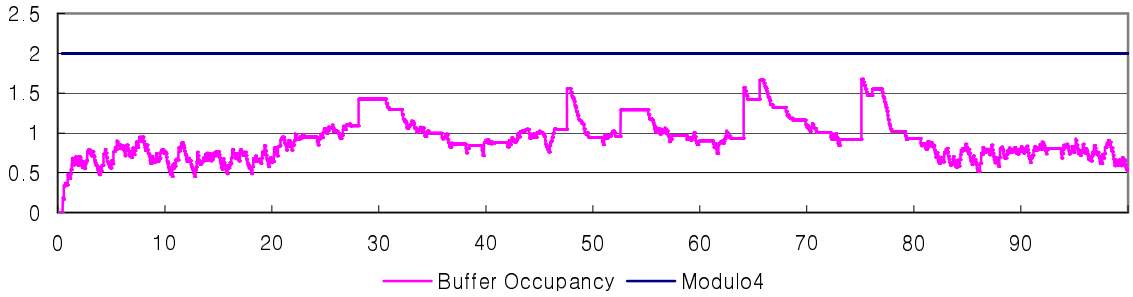
Through the following simulation where a single TCP connetion interfered with other contending CBR source, we can evaluate the beneficial use of the buffer occupancy and the contention factor in terms of signifying present medium contention degree.

Scenario: As seen below, a bulk FTP transfer is ongoing from node 0 to node 5 for 100 seconds with a contending traffic video source of CBR (packet size 512 bytes and interval

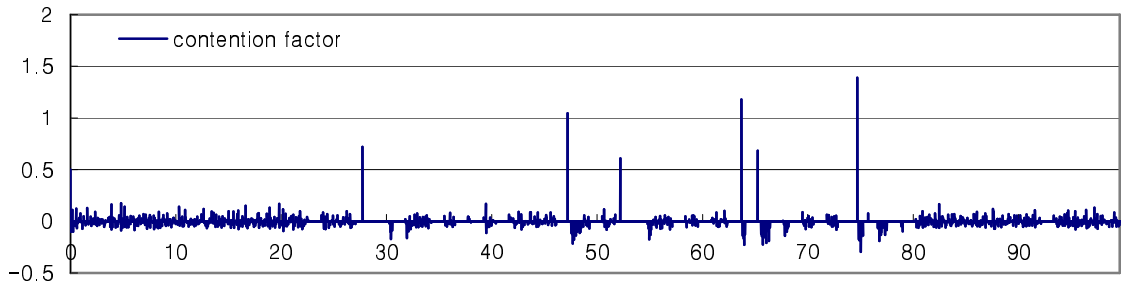
0.02, 200 Kbps) applied from node 6 to node 4, active 20 to 80 sec to interfere with the origination and reception of node 2, 3, 4, and 5.



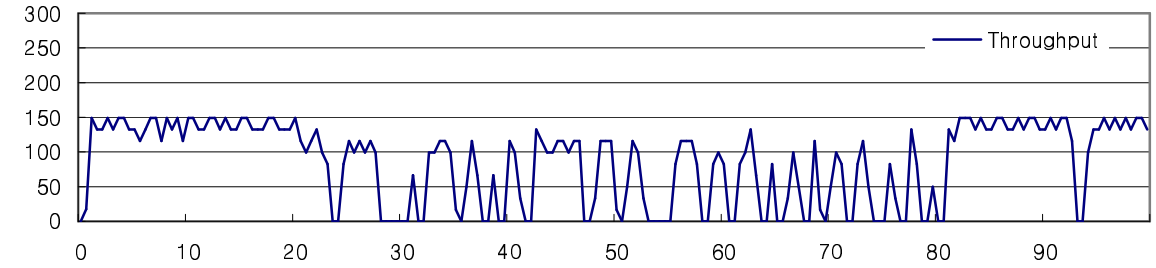
As shown in Figure 8-2 (c) and Figure 8.3 (c), the path link suffered from the medium contention in the period of 20 to 60 sec to a considerable extent. For the *ad hoc TCP*, as evident in Figure 8-2 (a) and (b), the medium contention was recognized appropriately by the buffer occupancy and the contention factor, and for the baseline, as evident in Figure 8-3 (a) and (b), ultimately packet forwarding was disabled in some part of the contention period that was also perceived by them.



(a) Buffer occupancy, monitoring the network buffer utilization, for this case can perceive the present medium contention once other competing source was present.

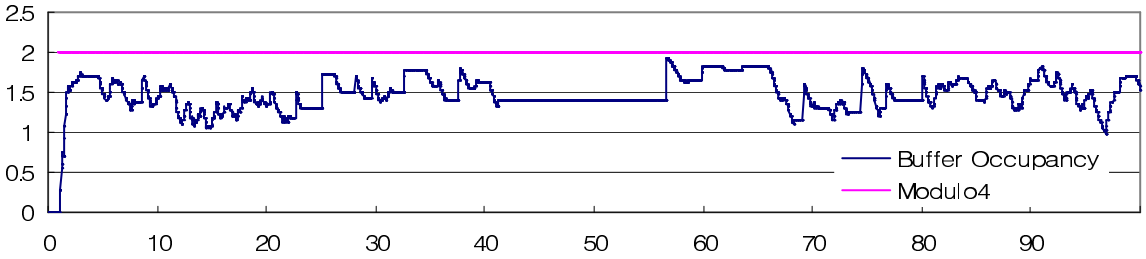


(b) Contention factor, can signify the instances of heavy contention present.

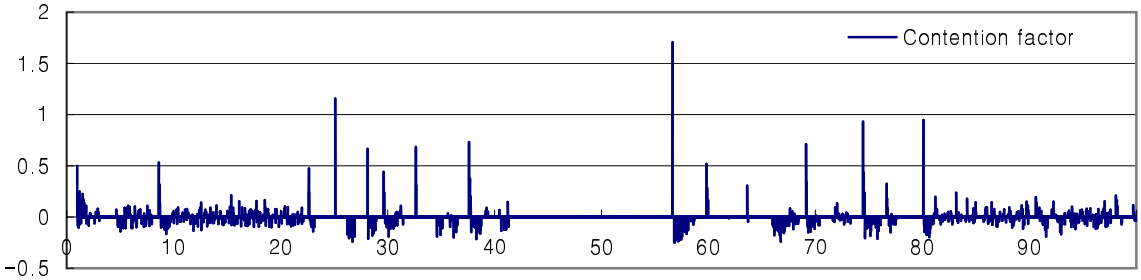


(c) Throughput obtained by the *ad hoc TCP*

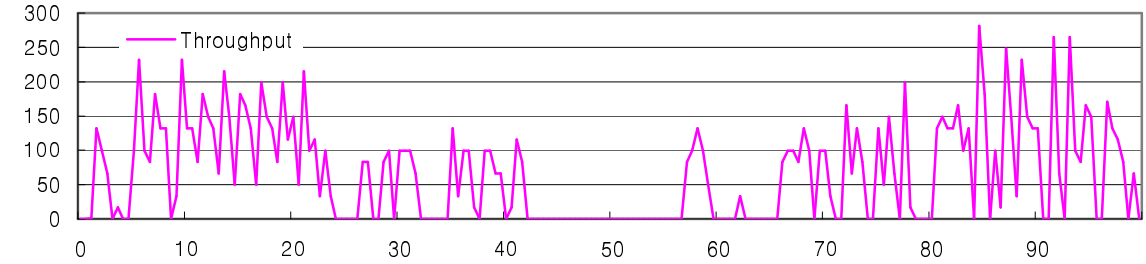
Figure 7-2. Medium contention perception according to the ad hoc TCP.



(a) Buffer occupancy, as expected, was leveled relatively higher than that of the *ad hoc TCP*, implying high network buffer utilization. For this case, the medium contention hardly signified, even though other competing source was present, because the baseline causes consistently high buffer utilization for the connection time.



(b) Contention factor, can still signify the instances of heavy contention present to an extent. In this sense, either freezing timer or *contention factor* may be used to throttle the sender's transmission window.

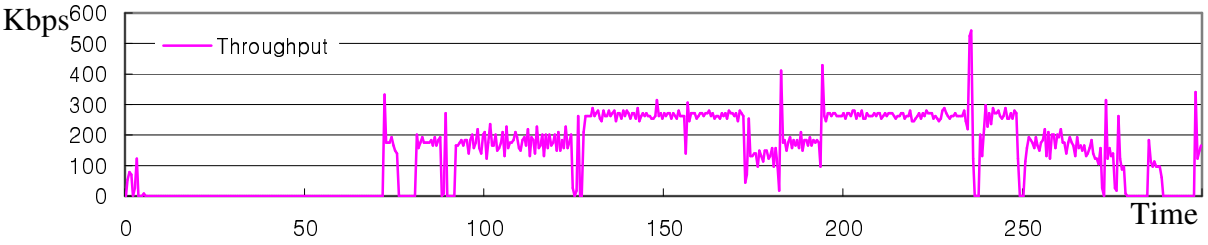


(c) Throughput obtained by the baseline.

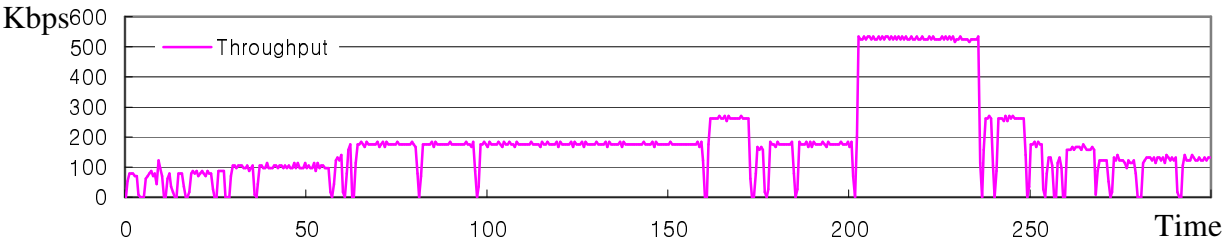
Figure 7-3. Medium contention perception according to the baseline

We can roughly conclude that the *buffer occupancy* and the *contention factor* can play role in signifying the change of the forward path link condition, specifically in terms of the network buffer utilization and the medium contention degree, respectively. Accordingly the receiver can throttle the sender's transmission window, or inform of the appropriate sending rate.

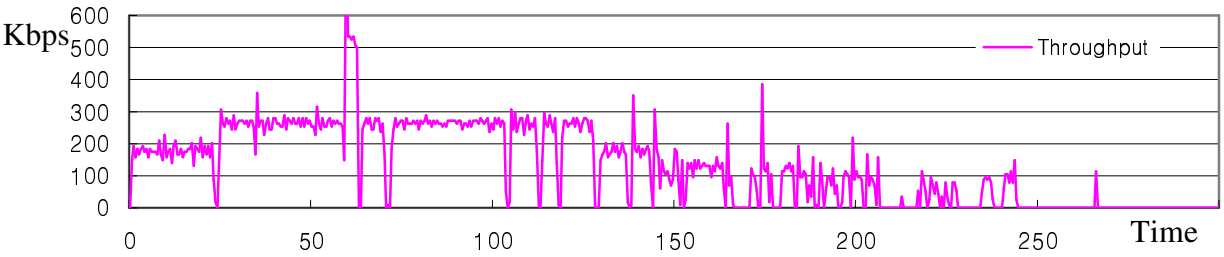
A.6 Throughput gain according to mobility changes in 1000 m x 1000 m with no other traffic.



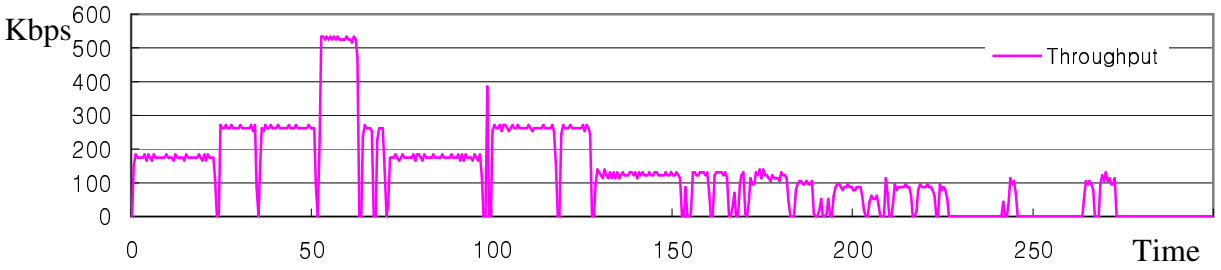
(a) Throughput measured at the receiver. Baseline at 10 m/s max. speed.



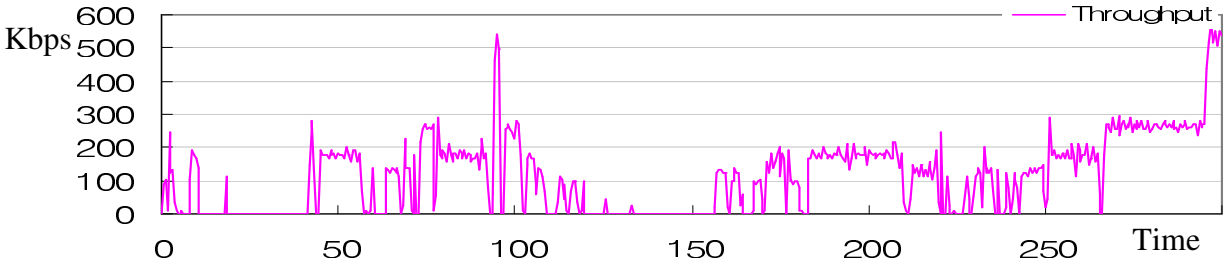
(b) Ad hoc TCP at 10 m/s.



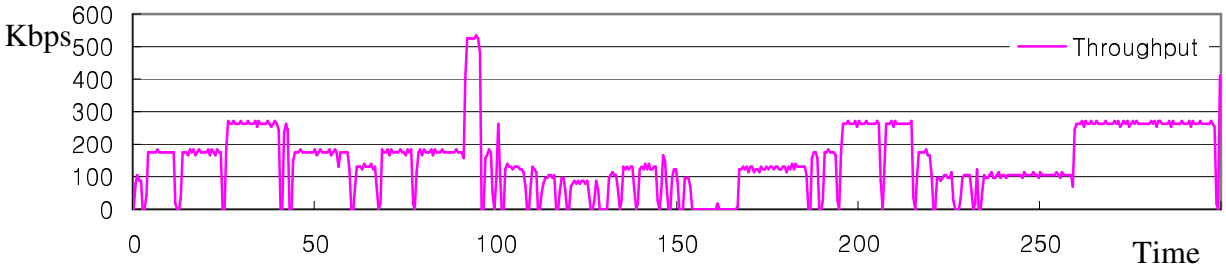
(c) Sack TCP at 20 m/s.



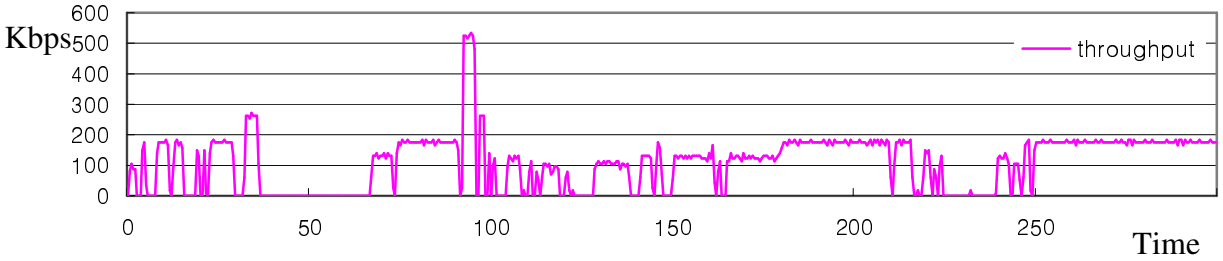
(d) Ad hoc TCP at 20 m/s.



(e) Sack TCP at 30 m/s.



(f) Ad hoc TCP at 30 m/s.



(g) Delimiter + freezer at 30 m/s.

Figure 7-4. Throughput gain by the ad hoc TCP in different max. node speeds. Different max. node speed outputs a different node movement pattern.

Given a node movement pattern of each of max. speed 10 m/s of average hop lengh of 5, 20 m/s of 4, and 30 m/s of 3, the ad hoc TCP outperforms the normal Sack TCP in terms of inefficient exponential backoffs and improper aggressive transmission rate as the normal TCP has. And as in

(g), the absence of ad hoc sender enhancements may cause the sender to suffer from the exponential backoffs sporadically due to sudden timeouts undetected as seen.

And, as (g) does, the use of *delimiter* with freezer implies that when connected to the wireline sender, the *delimiter* and freezer can control the sender by the ad hoc link capacity given that a rate calibration is made as in Appendix A.4.

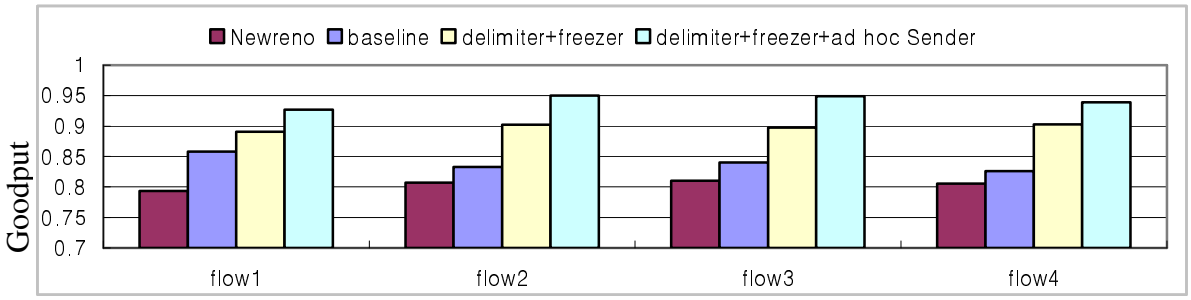
A.7 Differentiated Ad hoc Sender's behavior

This section explains how passively to discriminate an ad hoc receiver from a normal TCP receiver because it is necessary that the sender can behave differently according to the type of its opponent. Here is a way the ad hoc sender can perceive the other end. At an initial slow start phase, the sender usually increments by one every an ACK reception until it encounters a packet loss (perceived by either fast retransmit or timeout). Before a packet loss is occured, occasionally the sender could be limited by the receiver's controlled *adwin*. If so, then it could recognize that it is communicating with an ad hoc receiver for the current TCP session[1]. On the other hand, if it has never been under the *receive window-limited*, it is not likely to have been communicating with ad hoc one but with wireline TCP receiver that in general advertises much higher.

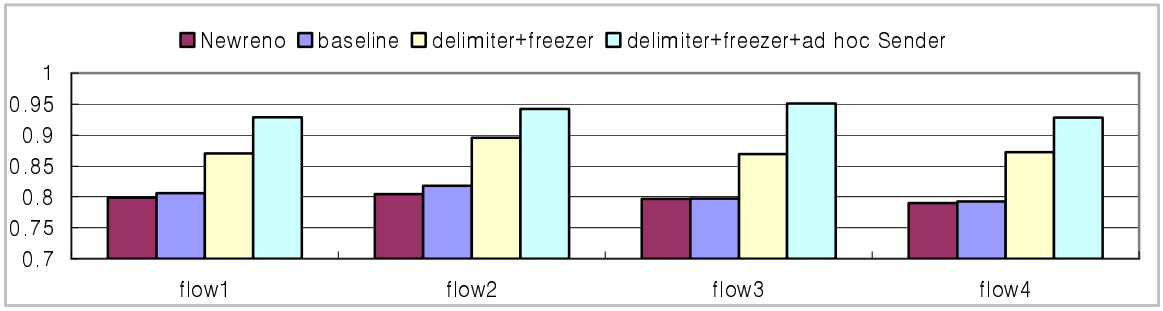
According to the passive determination, the ad hoc sender should differently set *ssthresh* each time a packet is retransmitted. If the sender perceives its opponent is a wireline receiver, it will not take into account the *adwin* as *ssthresh* but simply shrink the current transmit window as normal in case of fast retransmit and timeout.

A.8 Goodputs of four flows of a different TCP in change of node mobility

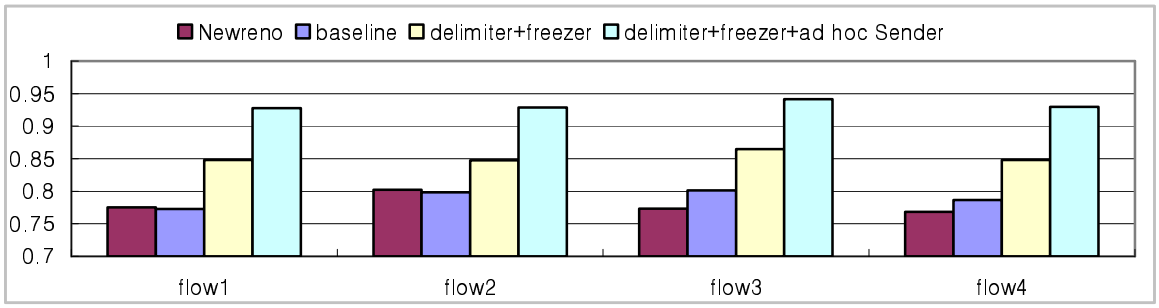
[1] Just in case, consider the wireline receiver might advertise less than congestion window thus limit. In order to avoid this ambiguity, the receiver can simply designate a bit in the option field to assure the connectivity between ad hoc ends.



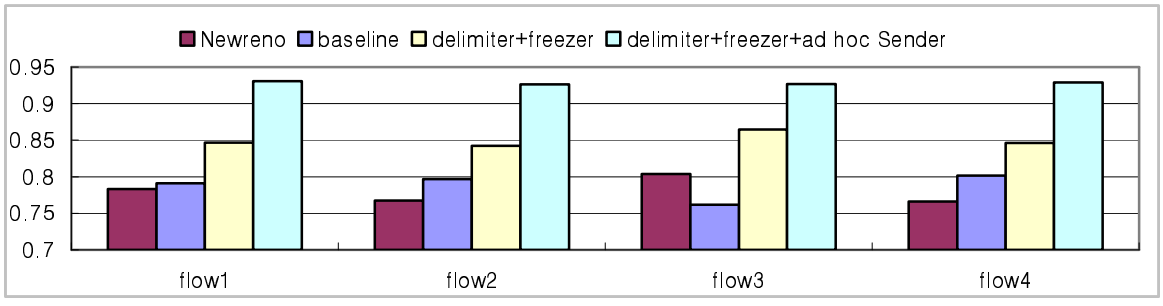
(a) Stationary



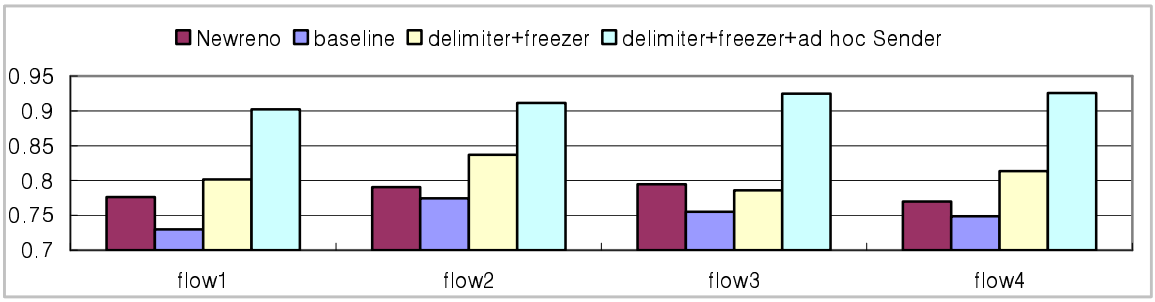
(b) 5 m/s



(c) 10 m/s



(d) 20 m/s



(f) 30 m/s

Figure 7-5. Goodputs of four flows of a different TCP in change of node mobility. Max. 24 % goodput gain at 30 m/s.

Appendix B. ns-2.1b9a tcl scripts and C++ codes used

(The CD-ROM contains the following files.)

B1. Tcl scripts

"adhocTCP.tcl"	–	main simulation tcl script outsourcing connection pattern and traffic scenario.
"data_ns_.rec"	-	metrics measurement for analyses
"traffic.tcl"	-	contains several traffic generating functions.

$sink_($tcp_cnt) set set_adwin_ 32 (advertised window set by the receiver)
$sink_($tcp_cnt) set rater_ 6 (the number of rater being used)

rater_ 1 :	rater
rater_ 4 :	congestion window delimiter with Equation 3-15
rater_ 5 :	only delimited by Equation 3-14
rater_ 6 :	intentional adwin by user with variable, set_adwin_
rater_ 7 :	delimited by Equation 3-16
rater_ 8 :	delimited by current hops (i.e., the upper bound of Equatin 3-17, Wpractical)

$sink_($tcp_cnt) set freezer_ 4 (the number of freezer)

freezer_ 1 :	freezer 1
freezer_ 2 :	freezer 2
freezer_ 3 :	both freezers
Otherwise,	no freezer

$sink_($tcp_cnt) set alpha_ 0.5 (freezer 1 threshold)
$sink_($tcp_cnt) set beta_ 5 (freezer 2 threshold)

B2. C++ codes

- tcp-sack1.cc - sender has four options to function

```
#define adhoc          // Probing capability    as in Chapter 3.5
#define adhocSender // sender's modification as in Chapter 3.5
#define provisional    // provisional frozen at sender as in section 3.5.1.3
#define rater          //only if rater is defined, the sender can take every adwin
                       //coming. If not, the sender takes a default wnd_ except ZWA
```

- adhocsink.h - the header file of “adhocsink.cc”
- adhocsink.cc - forms the ad hoc sink agent

set sink_($tcp_cnt) [new Agent/TCPSink/Sack1/AdHocSink] (the ad hoc sink agent)

- Add the line in “makefile” for generating object files

Bibliography

1. C. W Ahn, C. G. Kang, and Y. Z. Cho. Soft reservation multiple access with priority assignment (SRMA/PA): A novel MAC protocol for QoS-guaranteed integrated services in mobile ad hoc networks. In Proceedings of IEEE VTC 2000 spring, pages 942-947, Tokyo, Japan, May 2000.
2. G.S. Ahn, A.T. Campbell, A. Veres and L-H Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks", Proc. IEEE Infocom 2002, New York, New York, June 2002.
3. J.S.Ahn, P.B.Danzig et.al "TCP Vegas: New Techniques for Congestion Detection and Avoidance," SIGCOMM 94.
4. J.S. Ahn, Peter B. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: Emulation and Experiment, " In Proceedings of the ACM SIGCOMM, pp. 185--195, Cambridge, Massachusetts, Aug. 1995. ACM.
5. Aldar C.-F. Chany, Danny H. K. Tsangy, Sanjay Gupta, " Impacts of Handoff on TCP Performance in Mobile Wireless ," Department of Electrical and Electronic Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, HONG KONG.
6. Mark Allman, On the generation and Use of TCP acknowledgements, NASA Lewis Research Center/ Sterling Software, ACM Computer Communication Review, Oct 1998.
7. Mark Allman, Aaron Falk, "*On the Effective Evaluation of TCP*," ACM Computer Communication Review, 29(5), October 1999.
8. M. Allman, H. Balakrisnan, and S. Floyed, Enhancing TCP's Loss recovery using limited transmit, RFC 3042, Jan 2001.
9. M. Allman, S. Floyd, C Patridge, "Increasing TCP's initial Window," RFC3390, October 2002.
10. M. Allman and V. Paxson," On Estimating End-to-End Network Path Properties," Sigcomm 99.
11. M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control," RFC2581, April 1999.
12. A. Alwan, R. Bagrodia, N. Bambos, M. Gerla, L. Klenrock, J. Short, and J. Villasenor. Adaptive mobile multimedia networks. IEEE Personal Communications, 3(2):34-51, April 1996.
13. Elan Amir, Hari Balakrishnan, Srinivasan Seshan, Randy H. Katz – Efficient TCP over Networks with Wireless Links.
14. Ajay Bakre, B. R. Badrinath – I-TCP: Indirect TCP for Mobile Hosts –Oct 1994.

15. V. Anantharaman et al “A Microscopic Analysis of TCP performance over wireless ad hoc networks,”
16. J. Aweya, M. Ouellette, D.Y. Montuno and Zhonghui Yao, “Enhancing network performance with TCP rate control,” IEEE GLOBECOM ’00, vol.3, pp1712-1718, 2000.
17. Bikram S. Bakshi, P, Krishna, N.H. Vaidya, and D.K. Pradhan, “Improving Performance of TCP over wireless network,”
18. H. Balakrisnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, A comparison of mechanisms for improving TCP performance over wireless links, in IEEE/ACM Transactions on Networking, Vol. 5, No. 6, Dec 1997.
19. Hari Balakrisnan, Srinivasan Seshan and Randy H.Katz. *Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks*. Proceedings of MOBICOM'95, November 14-15, Berkeley, California, USA.
20. Hari Balakrishnan, Srinivasan Seshan, Elan Amir and Randy H. Katz, “Improving TCP/IP Performance over Wireless Networks,”.
21. Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan and Randy H. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links,”.
22. S. Banerjee and A. Agrawala, ”Estimating Available Capacity of a Network Connection,” In Proceedings of IEEE International Conference on Networks, September 2001.
23. D. Bansal, A. Chandra, and R. Shorey, “An extension of the TCP flow control algorithm for wireless networks,” in Proc. Of 1999 IEEE International Conference on Personal Wireless Communication, 1999.
24. Belding-Royer E.M, Sun Y, Perkin C, ”Global Connectivity for IPV4 Mobile ad Hoc Networks,” IETF Internet Draft, Nov. 2001. Work in Progress.
25. Brahim Bensaou, Yu Wang, Chi Chung Ko, “Fair Medium Access in 802.11 based wireless Ad-Hoc networks,” Centre for wireless communications, Singapore University,
26. D. Bertsekas and R. Gallager, “Data Networks,” Chapter 3.2, page 152. Prantice Hall, Second edition, 1992.
27. V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, “MACAW: A media access protocol for wireless LANs,” In Proceedings of ACM SIGCOMM ’94, 1994.
28. Saad Biaz and Nitin H. Vaidya, ”Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result,”.
29. S. Biaz and N. H. Vaidya, "*Discriminating congestion losses from wireless losses using inter-arrival times at the receiver*," IEEE Symp. ASSET'99, 1999.

30. Bikram S. Bakshi, P. Krishna, N. H. Vaidya, D. K. Pradhan – Improving performance of TCP over wireless networks.
31. E. Blanton, M. Allman, K. Fall, L. Wang, “A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP,” RFC 3517.
32. F. Bonomi and K.W. Fendick, “The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service,” IEEE Network, March/April 1995, pp. 25-39.
33. D. Borman, R. Braden, and V.Jacobson, “TCP extensions for high performance,” IETF RFC 1323, May 1992.
34. B. Braden et. al. , ”Recommendations on queue management and congestion avoidance in the Internet,” Apr. 1998. IETF Internet Request For Comments: RFC 2309.
35. R. T. Bradenl, Requirements for Internet hosts - communication layers, Internet RFC 1122, October 1989.
36. Brewer E. A., Katz R.H., Chawathe Y, Gribble S.D., Hodes T., Nguyen G., Stemm M., Henderson T., Amit E., Balakrishnan H., Fox A., Padmanabhan V., Seshan S (1998), “ A network architecture for heterogeneous mobile computing,” IEEE Personal Communications, 5 (5).
37. Josh Broch, David B. Johnson, David A. Maltz, The dynamic Source Routing Protocol for Mobile Ad hoc Networks, Internet-Draft, draft-ieft-manet-dsr-00.txt, March 1998.
38. K. Brown and S. Singh, "*M-TCP: TCP for mobile cellular networks*," ACM Computer Communications Review, vol. 27, pp. 19--43, Oct. 1997.
39. Kartik Chandran, Sudarshan Raghunathan, S. Venkatesan, Ravi Prakash – A Feedback Based Scheme For Improving TCP Performance In Ad-Hoc Wireless Networks.
40. A. Chockalingam, M. Zorzi and V. Tralli, "*Wireless TCP performance with link layer FEC/ARQ*", Proceedings of the ICC99, 1999, pp. 1212-1216.
41. A. Charny, K.K. Ramakrishnan, and A. Lauck, “Time Scale Analysis and Scalability Issues for explicit Rate Allocation un ATM Networks,” IEEE/ACM Trans. on Networking, 4(4), August 1996, pp.569-581.
42. Kai Chen and Klara Nahrstedt, EXACT: A Explicit Rate-based Flow Control Framework in MANET (extended version), University of Illinois at Urbana-Champaign, July, 2002.
43. David D. Clark. RFC-813: Window and Acknowledgement Strategy in TCP. Request For Comments, July 1982. Network Information Center.
44. David D. Clark and Fang W., "Explicit Allocation of Best Effort Packet Delivery Service," ACM Transactions on Networking, Aug 1998.

45. M.S. Corson and J. Macker, "*Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*," RFC 2501, Internet Engineering Task Force, Jan. 1999.
46. B. P. Crow, I. Widjaja, J. G. Kim and P. T. Sakai. IEEE 802.11 Wireless Local Area Network. IEEE Communications Magazine, September 1997.
47. R. Cunningham and V. Cahill, "Time bounded Medium Access Control for Ad Hoc Networks," in Principles of Mobile Computing (POMC'2002), Toulouse, France, October 30-31, 2002.
48. S. Dawkins, G. Montenegro, M. Kojo, and V. Magret. End-to end Performance Implications of Slow Links. IETF Internet draft "draft-ietf-pilc-slow-05.txt". November 2000.
49. T. D. Dyer, R. V. Boppana, A comparison of TCP performance over Three Routing protocols for Mobile Ad Hoc Netowrks. ACM Symposium on Mobile Ad Hoc Networking & Computing –Mobihoc, Oct 2001.
50. David A. Eckhardt and Peter Steenkiste – Improving Wireless LAN Performance via Adaptive Local Error Control.
51. Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi, "Signal Stability Based Adaptive Routing (SSA) for Ad Hoc mobile Networks," IEEE Personal Communications, Feburary 1997.
52. Elizabeth M. Royer, Charles E. Perins, An implementation Study of the AODV Routing Protocol
53. Kevin Fall, Sally Floyd, Simulation based Comparisons of Tahoe, Reno, and SACK TCP, Lawrence Berkely National Laboratory.
54. L. Feeney, "A taxonomy for routing protocols in mobile ad hoc networks," Technical Report T99/07, Swedish Institute of Computer Science, October 1999.
55. Fei P., Shiduan C., and Jian M., An effective way to improve TCP performance in wireless/mobile networks, in Proc. Of IEEE/AFCEA EUROCMM 2000, Information Systems for Enhanced Public Safety and Security, 2000.
56. Feng Wang, Yongguang Zhang, "Improving TCP performance over Mobile ad hoc networks with out-of-order detection and response," ACM, Mobihoc'02, June 2002.
57. Fu Chengpeng, "TCP veno: End-to-End Congestion Control over heterogeneous Networks," Ph.D dissertation, Chinese university of Hong Kong, July 2001.
58. Fu. Z., B. Greenstein, X, Meng, S. Lu. "Design and Implementation of a TCP-Friendly Transport Protocol for Ad Hoc Wireless Network," 10th IEEE Internation Conference on Network Protocols (Paris, France), IEEE, 212-225, 2002.

59. C. Fullmer and J. J. Garcia-Luna-Aceves, “Floor Acquisition Multiple Access (FAMA) for packet radio networks”, Computer Communication Review, vol. 25, (no.4), ACM SIGCOMM ’95, Cambridge, MA, USA, 28 Aug.-1 Sept. 1995, Oct. 1995.
60. Sally Floyd, A Report on Recent Deleopments in TCP congestion control. IEEE Communications Magazine, April 2001.
61. S. Floyd, T. Henderson, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” Network Working Group, RFC 2582, April 1999.
62. S. Floyd, J. Mahdavi, M. Mathis, M podolsky, An extension for the Selective Acknowledgement (SACK) option for TCP, RFC2883.
63. Zhenghua Fu, Peros Zerfos, Kaixin Xu, Haiyun Luo, Songwu Lu, Lixia Zhang, Mario Gerla, “On TCP performance in Multihop Wireless Networks,” university of California at Los Angeles.
64. Vahid Garousi, “Simulationg Network traffic in Multihop wireless Ad Hoc Networks based on DSDV protocol using NS package,” University of Wasterloo, project report, 2001, available from : <http://www.cst.uwaterloo.ca/~garousi/downloads/750-proj-rep.pdf> [Accessed 5 Jan 2004].
65. Mario Gerla, Ken Tang, Rajive Bagrodia - TCP Performance in Wireless Multi-hop Networks.
66. M.Gerla, Wenkjie Wen and R.Lo Cigno, “Bandwidth feedback control of TCP and real time sources in the internet,” IEEE GLOBECOM ’00, vol.1, pp.561-565, 2000.
67. S.Goel, and d.Sanghi, Imroving TCP performance over wireless links, In Proc. Of TENCON’98, 1998 IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control, Vol.2, Dec 1998.
68. Tom Goff et al. “Freeze-TCP: A true end to end TCP enhance mechanism for mobile environments”
69. Tom Goff et al. Preemptive routing in ad hoc network.
70. Tom Goff et al. Analysis of TCP performance on Ad hoc network using preemptive maintenance routing.
71. Marc Greis, Tutorial for the network simulator, VINT group, available from : < http://www.isi.edu/nsnam/ns/tutorial/index.html> [Accessed 1 Jan 2004].
72. Alex Ali Hamidian, “A Study of Internet Connectivity for Mobile Ad Hoc Networks in NS2,” Technical report, Lund Institute of Technology, Lund University, January 2003.
73. Xiao Hannan, “A Flexible Quality of Service Model for Mobile Ad Hoc Networks,” Ph.D. dissertation, Singapore University, Singapore, 2002.
74. Gavin Holland and Nitin Vaidya – “Analysis of TCP Performance over Mobile Ad-Hoc networks”.

75. The Institute of Electrical and Electronics Engineer, inc. IEEE std 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. The Institute of Electrical and electronics Enginner, inc., 1999 edition.
76. Jacobson V., "Congestion Avoidance and Control," Computer Communication Review, vol. 18, no. 4, August 1988.
77. V. Jacobson, "*Modified TCP congestion avoidance algorithm*", end2end-interest mailing list, April 1990.
78. Manish Jain and Constantinos Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in Proceedings of the ACM SIGCOMM 2002.
79. H. Jiang and C. Dovrolis,"*Passive estimation of TCP round-trip tim*es," ACM Computer Communication Review, pages 75--88, July 2002.
80. D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in Mobile Computing, edited by T. Imielinski and H. Korth, Chapter 5, pp.153-181, Kluwer Academic Publishers, 1996
81. L. Kalampoukas, A. Varma and K. K. Ramakrishnan, Explicit Window adaptation: a method to enhance TCP performance. In Proeeding of IEEE Infocom'98, vol.1, pp.242-251, 1998.
82. P. Karn. MACA – a new channel accesss method for packet radio. In ARRL/CRRL Amateur Radio 9th Computer Networking Conference, pages 134-140, 1990.
83. Karn P. and C. Partridge, "Improving Round-trip time estimates in Reliable Transport Protocols", SIGCOMM 87.
84. A. Karnik and A. Kumar,"Performance of TCP Congestion Control with Explicit Rate Feedback: Rate Adaptive TCP (RATCP)," IEEE GLOVECOM '00, vol.1, pp571-576, 2000.
85. R.H. Katz – Adaptation and Mobility in Wireless Information Systems – IEEE Personal Communications, 1 (1), 1994.
86. Dongkyun Kim et al. "TCP-bus: Improving TCP performance in ad hoc networks"
87. S. Lee and A. T. Campbell, "*INSIGNIA: In-band signaling support for QoS in mobile ad hoc networks*," in Proc. of the 5 th International Workshop on Mobile Multimedia Communications, 1998.
88. Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris, "Capacity of Ad Hoc Wireless Network," MIT laboratory, computer Science,
89. Young shying Liaw, Qinghe Yin, Joshua Liew, Winston seah –Decoupling of TCP Congestion control and loss recovery for wireless network.

90. C. R. Lin and M. Gerla. Asynchronous multimedia multihop wireless networks. In Proceedings of IEEE INFOCOM '97, pages 118-125, Kobe, Japan, April 1997.
91. Jian Liu and Suresh Singh – ATCP: TCP for Mobile Ad Hoc Networks.
92. R. Ludwig, "A case for flow--adaptive wireless links," Technical report, University of California, Berkeley, 1999.
93. R. Ludwig, "Eliminating inefficient cross-layer interactions in wireless networking," Ph.D. dissertation, Technische Hochschule Aachen, Germany, April 2000.
94. Ludwig, R., and Katz, R. The Eifel Algorithm: Making TCP robust against spurious retransmissions. ACM Computer Communications Review 30, 1, Jan. 2000.
95. Ludwig R., Rathonyi B., Konrad A., Oden K., Joseph A., "Multi-Layer Tracing of TCP over a Reliable Wireless Links," In Proceedings of ACM SIGMETRICS 99.
96. David A. Maltz, Josh Broch and David B. Johnson. Experiences designing and building a multi-hop wireless and ad hoc network test bed. Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, March 1999.
97. David A. Maltz et el. Lessons form a full scale Multihop Wireless Ad hoc Network Testbed.
98. J. Macker and S. Corson. Mobile ad hoc networks(MANET). http://www.ietf. org/html.charters/manet-charter.html, 1997. IETF Working Group Charter.
99. S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, Ren Wang, "*TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks*," ACM Mobicom 2001.
100. M. Mathis and J. Mahdavi, "Forward Acknowledgement: Refining TCP Congestion Control," SIGCOMM 96, August 1996
101. M.Mathis et el. TCP Selective Acknowledge Options, RFC2018.
102. D. Mills, "Network time protocol (version 3): Specification, implementation, and analysis," Tech. Rep., Network Information Center, SRI International, Menol Park, CA, March, 1992.
103. Mogul, J. C., Deering, S. E., " Path MTU Discovery," RFC 1191, November 1990.
104. J. P. Monks, P. Sinha, and V. Bharghavan, "*Limitations of TCP-ELFN for ad hoc networks*," in Proc. of The 7th Int'l Workshop on Mobile Multimedia Communications, MoMuC 2000.
105. S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," Proc. of INFOCOM '99, pp. 227-234, Mar. 1999.
106. Ns-Network Simulator, the VINT project at LBL, officially available from: http://www.isi.edu/nanam/ns [Accessed Jan 2004].

107. Ns-2 C++ Class Hierarchy, publicly available from: http://www.isi.edu/nsnam /nsdoc-classes/hierarchy.html [Accessed Jan 2004].

108. Ns by example, "OTcl Linkage," available from: http://nile.wpi.edu/ NS/linkage.html [Accessed Jan 2004].

109. Ruy de Oliveira and Torsten Braun, TCP in Wireless Mobile Ad Hoc Networks, Berne university, Switzerland, 2002

110. Ruy de Oliveira, Torsten Braun, Marc Heissenbuttel, "An Edge-based Approach for Improving TCP in Wireless Mobile Ad Hoc Networks," Berne University, Switzerland, 2002.

111. OTcl Tutorial, available from: http://cvs.sourceforge.net/viewcvs.py/otcl-tclcl/otcl/doc/tutorial.html?rev=HEAD [Accessed Jan 2004].

112. C. Parsa and J.J. Garcia-Luna-Aceves, "*Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media*", In Proceedings of IEEE ICNP '99, Toronto, October 1999.

113. C. Parsa and J. J. Garcia-Luna-Aceves, "*TULIP: A Link-Level Protocol for Improving TCP over Wireless Links*," Proc. IEEE WCNC'99, pp. 1253--1257, 1999.

114. Shyam Pather, Introduction to Programming with Tcl, available at http://hegel.ittc.ukans.edu/topics/tcltk/tutorial-noplugin.html [Accessed Jan 2004].

115. V.Paxson, M. Allman, "Computing TCP's Retransmission Timer," RFC 2988, Nov., 2000.

116. Charles E. Perkins, Jari T. Malinen, Ryuji Wakikawa, Anders Nilsson, and Antti J. Tuominen, "Internet Connectivity for Mobile ad hoc networks," In Wireless Communications and Mobile Computing, 2:465-482, 2002.

117. Charles E. Perkins, Elizabeth M. Belding-Royer, Samir R. Das, Ad hoc On-Demand Vector (AODV) routing, Internet draft, 19 June 2002.

118. Dmitri D. Perkins and Herman D. Hughes, "The interaction of Medium Access Control and TCP in Ad Hoc networks," The International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SanDiego, California, July 14-18, 2002.

119. Larry L. Peterson, Bruce S. Davie, "Computer Networks—A System Approach," Morgan Kaufmann Inc, p 22-29, 1996.

120. N. Poojary, S. V. Krishnamurthy and S. Dao., "Medium Access Control in a Network of Ad Hoc Mobile Nodes with heterogeneous Power Capabilities," In proceedings of ICC2001, June 2001.

121. J. Postel, ed., "User Datagram Protocol; RFC-768,"inin Internet Requests for Comments,No. 768. August 1980.

122. J. B. Postel, Transmission Control Protocol, RFC, Information Sciences Institute, Marina del Rey, CA, September 1981. RFC-793.

123. Qiang Ni, Thierry Turletti, Wei Fu, “Simulation-based Analysis of TCP Behavior over Hybrid Wireless Wired Networks,” 1st International Workshop on Wired/Wireless Internet Communications (WWIC 2002).

124. K. Ramakrishnan, S. Floyd, D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP,” RFC 3168, September 2001.

125. Venkatesh Ramarathinam and Miguel A. Labrador, “Performance Analysis of TCP over Static Ad Hoc Wireless Networks,” In Proceedings of the ISCA 15th International Conference on Parallel and Distributed Computing Systems (PDCS)", Pages 410-415, September 2002.

126. K. Ratnam, and I. Matta, "*WTCP: An efficient transmission control protocol for networks with wireless links*," IEEE Symp. Computer and Communications (ISCC), June 1998.

127. E. M. Royer, Y. Sun, and C. Perkins, "*Global Connectivity for IPv4 Mobile Ad hoc Networks*," in draft-ietf-manet-globalv4-00.txt, November, 2001.

128. Elizabeth M. Royer and C.-K. Toh, "*A review of current routing protocols for ad-hoc mobile wireless networks*," IEEE Personal Communications Magazine, pp. 46-55, April 1999.

129. Ray Saikat, Jeffrey B. Carruthers and Starobinski David, RTS/CTS-Induced Congestion in Ad Hoc Wireless LANs, IEEE WCNC 2003, New Orleans, March 2003, pp. 1516-1521.

130. N.K.G Samaraweera, Non-congestion packet loss detection for TCP error recovery using wireless links, IEE proceedings –Communications, Vol.146, No.4, Aug 1999.

131. J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP buffer tuning," in Proceedings of ACM SIGCOMM'98, pp. 31523, Aug. 1998.

132. Seung-Joon Seok, Sung-Bum Joo, and Chul-Hee Kang, A Mechanism for Improving TCP Performance In Wireless Environments, Korea University, Seoul, Korea.

133. Shih-LinWu, et al, “ Route maintenance in a wireless mobile ad hoc network”

134. P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bhargavan WTCP: A Reliable Transport Protocol for Wireless Wide-area Networks. TIMELY Group Research Report, January 1999.

135. W. Richard Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, Jan 1997, RFC 2001.

136. W. Richard Stevens, Mark Allman, Vern Paxson, TCP congestion Control, August 1998, Internet-Draft draft-ietf-tcpimpl-cong-control-00.txt.

137. Dong Sun, Hong Man, “Evaluation of Reliable Data Transport Schemes for Mobile Ad Hoc Networks ,”

138. D. Sun and H. Man, "*Performance comparison of transport control protocols over mobile ad hoc networks*," in The 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC), (San Diego, CA), Sep. 2001.

139. Karthikeyyan Sundaresan, Vaidyanathan Anantharaman, Hung-Yun Hsieh, and Raghupathy Sivakumar, ATP: A Reliable Transport Protocol for Ad-hoc networks, MobiHoc'03, Jun, 2003.

140. Z. Tang and J. J. Garcia-Luna-Aceves. Hop-reservation multiple access (HRMA) for ad hoc networks. In Proceedings of IEEE INFOCOM '99, pages 194-201, 1999.

141. Z. Tang and J. J. Garcia-Luna-Aceves. A protocol for topology dependent transmission scheduling in wireless networks. In Proceedings of WCNC '99, pages 1333-1337, 1999.

142. Ken Tang, Mario Correa and Mario Gerla, Effects of Ad Hoc MAC layer Medium Access Mechanisms under TCP, University of California, Los Angeles.

143. K. Tang and M. Gerla, "Fair Sharing of MAC under TCP in wireless Ad Hoc Networks," In Proceedings of IEEE MMT '99, October 2000.

144. The ATM forum, Traffic Management Specification (Version 4.1), document AF-TM-0121.000, March, 1999, pp.43-55.

145. J. Tsai and M. Gerla. Multiple mobile multimedia radio network. ACM-Baltzer Journal of Wireless Networks, 1(3):255-265, 1995.

146. Tihjia Tsai, Shiann-Tsong Sheu, Jenhui Chen, and Hou-Han Chen, "An Efficient Real-time MPEG-4 Transmission Scheme for Wireless Networks," Tamkang University, Tamsui, Taiwan.

147. V. Tsaoussidis and H. Badr, "TCP-Probing: Toward an Error Control Schema with Energy and Throughput Performance Gains," Proc. 8th IEEE Conf. Network Protocols, Japan, Nov. 2000.

148. V. Tsaoussidis and I. Matta, "Open Issues on TCP for Mobile Computing," In the Journal of Wireless Communications and Mobile Comupting, Wiley Academic Publishers, Issue 2, vol.2, Feburary 2002.

149. V. Tsaoussidis and C. Zhang, "TCP-Real: Receiver-oriented Congestion Control," The Journal of Computer Networks, 2002.

150. N. H. Vaidya, M. Mehta, C. Perkins, G. Montenegro, "*Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless*," Technical Report 99-003, Computer Science Dept., Texas A&M University, February 1999.

151. Z. Wang and J. Crowcroft, "A New Congestion Control Scheme: Slow Start and Search (Tri-S)," ACM Computer Communication Review, vol. 21, no. 1, Jan. 1991.

152. Haitao Wu, Yong Peng, Keping Long, Shiduan Cheng, and Jian Ma, "Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement," In Proceedings of the IEEE INFOCOM '02, June 2002.

153. S. Xu and T Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc network?," IEEE Communications Magazine, Vol 39, No. 6, pages 130-137, June 2001.

154. S. Xu and T. Saadawi, "Evaluation for TCP with Delayed ACK option in wireless multihop networks," In Proceeding of Semiannual Vehicular Technology Conference (VTC2001 Fall), October 2001.

155. S. Xu, T Saadawi and M. Lee, "On TCP over Wireless Multihop networks," In Proceedings of IEEE MILCOM2001, October 2001.

156. George Xylomenos and George C. Polzos, "TCP and UDP performance over a wireless LAN," In Proceedings of the IEEE INFOCOM '99, pages 439-446, March 1999.

157. Luqing yang, Winston K. G. Seah, Qinghe Yin, Improving Fairness among TCP Flows crossing Wireless Ad Hoc and Wired Networks

158. Wing Ho Yuen, Heung-no Lee, Timothy D. Andersen, "A simple and effective Cross Layer Networking System for Mobile Ad Hoc Networks,"

159. C. Zhu and M. S. Corson. A five-phase reservation protocol, (FPRP) mobile ad hoc networks. In Proceedings of IEEE INFOCOM'98, pages 322-331, 1998.

160. Michele Zorzi and Ramesh R. Rao – Error Control in Multi-Layered Stacks.